Lookout™ *Direct*

Object Reference Manual

AUTOMATION DIRECT.com

License Agreement

"Valued Technology"

This License Agreement is your proof of license. Please treat it as valuable property.This is a legal agreement between you (either an individual or entity) the end-user of this product, and Automation Direct. If you do not agree to the terms of this Agreement, promptly return the package and any accompanying items to Automation Direct for a full refund.

LookoutDirect SOFTWARE LICENSE

1. GRANT OF LICENSE — This LookoutDirect License Agreement ("License") permits you to use one copy of the specified version of the LookoutDirect software product ("Software") on any single computer, provided the Software is in use on only one computer at any time. If you have multiple Licenses for the Software, then at any time you may have as many copies of the Software in use as you have Licenses. The Software is deemed "in use" on a computer when it is loaded into the temporary memory (i.e. RAM) or installed into the permanent memory (i.e. hard disk, CD--ROM or other storage device) of that computer, except that a copy may be installed on a network server for the sole purpose of distribution to other computers that are not currently "in use". If the anticipated number of Software users will exceed the number of applicable Licenses, then you must have a reasonable mechanismor process in place to assure that the number of persons using the Software concurrently does not exceed the number of Licenses. If the Software is permanently installed on the hard disk or other storage device of a computer (other than a network server) and oneperson uses that computer more than80%of the timeit is in use, then that person may also use the software on a single portable computer or a single computer at home.

2. COPYRIGHT —The Software is owned by Automation Direct or its suppliers and is protected by United States copyright laws and international treaty provisions. Therefore, you must treat the Software like any other copyrighted material (e.g., a book or musical recording) except that you may:

   a. Make one backup copy of the Software solely for backup or archival purposes.

   b. Transfer the Software to a single hard disk provided you keep the original and the backup solely for archival purposes. You may not copy any written materials that may accompany the Software.

3. OTHER RESTRICTIONS—This LookoutDirect License Agreement is your proof of license toexercise therights grantedherein andmust beretained by you. You may not rent or lease the Software, but you may transfer your rights under this LookoutDirect License Agreement on a permanent basis provided that you transfer this License Agreement, the Software, and all accompanying written materials and retain no copies. The recipient must also agree to unequivocally abide by the terms contained in this License Agreement. You may not reverse engineer, decompile, or disassemble the Software. Any transfer of the Software must include the most recent update and all prior versions. If the Software is acquired within the United States, you may not export the Software outside of the United States without first complying with all applicable USexport laws and regulations. You acknowledge that the Software will not function without a certain hardware key. This hardware key will be furnished to you by Automation Direct and you agree that such hardware key is to be used solely with the Software provided.

DISTRIBUTION LIMITATIONS

If you have acquired the development/runtime package, you may distribute process files created with the Software, provided that:

1. each recipient of your process file has a valid license for a separate runtime copy of the Software;

2. you include the following copyright notice, either on--screen in your process file's About Box or written documentation, with each distributed copy of your

3. process file: “Copyright E [year] Automation Direct by Koyo, Inc. (Based on materials of National Instruments Corporation). All Rights Reserved”;

4. you do not use Automation Direct’s or National Instruments’ (”NI”) names, logos, or trademarks to market your process file without written permission; and

5. you agree to indemnify, hold harmless, and defend Automation Direct and NI (including their officers, directors, employees, and agents) and their suppliers fromand against any claims.

LIMITED WARRANTY

1. Automation Direct warrants the Software will perform substantially in accordance with the written materials for a period of ninety (90) days from the date of receipt.

2. Automation Direct warrants that any hardware accompanying the Software will be free fromdefects in materials and workmanshipunder normal use andservice for a period of one (1) year from the date of receipt.

NO OTHER WARRANTIES. EXCEPT AS EXPRESSLY SET FORTHABOVE, THE SOFTWARE IS PROVIDED“AS IS” WITHOUT WARRANTYOF ANYKIND, AND NO OTHER WARRANTIES, EITHER EXPRESSED OR IMPLIED ARE MADE WITH RESPECT TO THE SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NON--INFRINGEMENT, OR ANY OTHER WARRANTIES THAT MAY ARISE FROM USAGE OR TRADE OR COURSE OF DEALING. Automation Direct AND ITS SUPPLIERS DO NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE OF OR THE RESULTS OF THE USE OF THE SOFTWARE IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE AND DO NOT WARRANT THAT THE OPERATION OF THE SOFTWARE WILL BE

UNINTERRUPTED OR ERROR FREE. Automation Direct AND ITS SUPPLIERS EXPRESSLY DISCLAIM ANY WARRANTIES NOT STATED HEREIN.

Customer Remedies —Automation Direct’s entire liability and your exclusive remedy shall be at Automation Direct’s option. Automation Direct will either refund the price paid, or repair or replace the Software or hardware that does not meet Automation Direct’s Limited Warranty. You must return the product to Automation Direct with a copy of your purchase receipt. This Limited Warranty is void if failure of the Software or hardware has resulted from accident, abuse, or misapplication. Any replacement Software will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer.

No Other Warranties — Automation Direct DISCLAIMS ALL OTHER WARRANTIES, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WITH RESPECT TO THE SOFTWARE, AND ANY ACCOMPANYING WRITTEN MATERIALS OR HARDWARE. THIS LIMITED WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE OTHERS, WHICH VARY FROM STATE TO STATE.

No Liability for Consequential Damages — In no event shall Automation Direct or its suppliers be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use of or inability to use this Automation Direct product, even if Automation Direct or its suppliers have been advised of possibility of such damages.

U.S. GOVERNMENT RESTRICTED RIGHTS

The Software and documentation are provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227--7013 or subparagraphs (c) (1) (2) of the Commercial Computer Software -- Restricted Rights at 48 CFR

52.227--19 as applicable. Contractor/manufacturer is Automation Direct by Koyo, Inc. / 3505 Hutchinson Road, Cumming, GA 30040 (under license from National Instruments Corporation, 11500 N. Mopac Expressway, Austin, Texas 78759--3504).

This agreement is governed by the laws of the State of Georgia.

WARNING

(1) THE SOFTWARE IS NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR INCONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTEROPERATINGSYSTEMSOFTWARE FITNESS, FITNESS OF COMPILERS ANDDEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED ”SYSTEM FAILURES”). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK--UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END--USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM UTILIZED TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE THE SOFTWARE IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY PLCDIRECT OR ITS SUPPLIERS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF THE SOFTWARE WHENEVER THE SOFTWARE IS INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

LookoutTM Direct
Getting Started Guide

Please include the Manual Name and the Manual Issue, both shown below, when communicating with Technical Support regarding this publication.

Manual Name:	**LookoutDirect Object Reference**
Issue:	**First Edition, Rev. A**
Issue Date:	**11/02**

Publication History

Issue	Date	Description of Changes
First Edition	10/01	Orginal
Rev. A	11/02	Deleted information on Historgram object

Contents

About This Manual

Chapter 1
Introduction to the LookoutDirect Object Class Reference

Chapter 2
System Objects

Chapter 3
Driver and Protocol Objects

About This Manual

This manual describes Lookout*Direct* object classes, listed in alphabetical order, in two sections. The information contained in this manual can also be viewed in online help.

Document Conventions

The following document conventions are used in this manual.

»	Indicate the path for nested menu and command selections. Example: **Start » Programs » xxx**
💡	Indicates a tip that provides helpful information related to the current procedure or topic.
📝	Indicates a important supplementary information pertinent to the current procedure or topic.
⚠	Denotes a caution statement. Failure to follow the guidance provide in the caution statement could result in a loss of data or an interruption to a critical process being controlled or monitored by Lookout*Direct*.
bold face text	Denotes the name of dialog boxes, property sheet items, application features, and menu commands that you select as part of a procedure.
italic	Denotes references to dialog boxes, property sheet items, application features, and menu commands that are not part of the current procedure, or key concepts.
`monospace`	Denotes characters that you enter from the keyboard, code/syntax examples. The monospace font is also used to express proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames and extensions.

Technical Support

Phone

Contact Automation Direct's technical support department toll free at 1-770-844-4200. Technical support is available weekdays from 9:00 a.m through 6:00 p.m. EST.

World Wide Web

To access a wide variety of technical support assets including print-on-demand documentation and software downloads, visit the Automation Direct website at *www.automationdirect.com* and click on *Technical Support*.

1

Introduction to the Lookout*Direct* Object Class Reference

This manual describes Lookout*Direct* object classes, listed in alphabetical order, in two chapters.

Chapter 2, *System Objects*, covers the Lookout*Direct* System object classes, native to Lookout. You use objects made from these classes as controls and for data analysis and display.

Chapter 3, *Driver and Protocol Objects*, contains Lookout driver and protocol objects that you use to connect to PLCs, RTUs, and other hardware that is a part of your industrial automation and control system.

Input parameter syntax and data members are documented for each object class, along with a description of the functionality of each object class.

Note Lookout assists you in building graphical screen displays when possible. When you place a displayable object (like Pots or Switches) on a control panel, the appropriate **Display Parameter** dialog box appears, prompting you to select a display type. If the object is not displayable but supports an implicit signal (like Counters, LatchGates, and OneShots), Lookout inserts an expression on the control panel.

You can elect not to display an object or its signal by clicking on the **Cancel** button at any time. If you change your mind later, you can display the object or its signal using the **Insert»Displayable Object** or **Insert»Expression** commands, respectively.

Some object classes have neither a display member nor an implicit signal—instead they have multiple data members. If you want to display the result of a logical or numeric data member on a control panel, you can use the **Insert»Expression** command and choose the appropriate name and data member, or drag and drop the data member from the Lookout Object Explorer.

See the *Getting Started with Lookout* manual for more information on creating objects, modifying their databases, and linking them together.

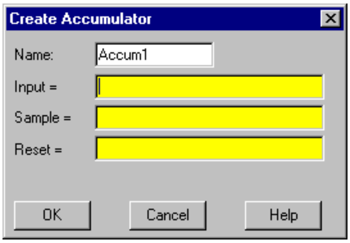
2

System Objects

This chapter describes Lookout*Direct* System object classes, listed in alphabetical order. Input parameter syntax and data members are documented for each object class, along with a description of the functionality of each object class.

Accumulator

Accumulator is a numeric totalizer. It samples the numeric **Input** any time the **Sample** value transitions from off to on, adding each new sample to the previous running total. It resets the totalized value to zero when **Reset** transitions from off to on.



Create Accumulator

Name: Accum1

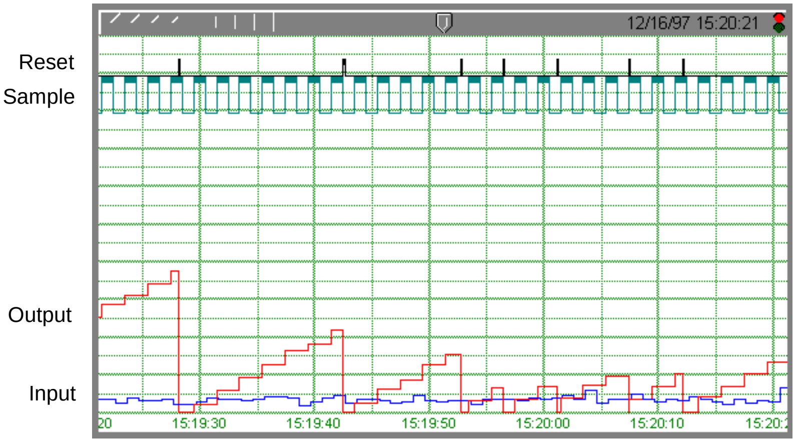
Input =

Sample =

Reset =

Input is a numeric expression while **Sample** and **Reset** are logical expressions.

The following HyperTrend demonstrates the relationship between **Input**, **Sample**, and **Reset**.



Accumulator Data Members

Table 2-1. Accumulator Data Members

Data Members	Type	Read	Write	Description
(implicit)	numeric	yes	no	Current totalized value, totalized since the most recent Reset signal. Updated at the defined Sample rate.

Comments **Reset** could be a regular pulse interval created by a Pulse timer, as shown in the dialog box in Figure .

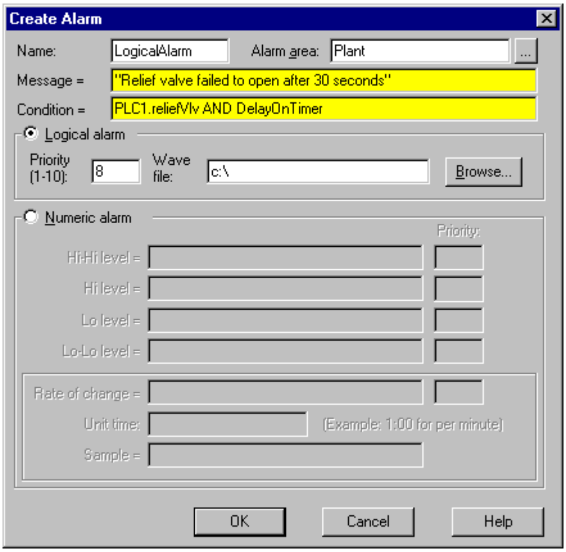
The Counter object class totalizes a number of logical events and the Integral object class totalizes rates. Use Accumulator to totalize numeric variables.

Related Objects *Average, Minimum, Maximum, Sample*

Alarm

Alarm is a flexible and powerful object class you use to create various logical- and numeric-triggered alarms that are displayed in the alarm window.

Note You should first read about the Lookout*Direct* alarm system in Chapter 9, *Alarms*, in the *LookoutDirect Developer's Manual* to aid you in designing the most efficient alarming scheme for your application.



Create Alarm

Name: LogicalAlarm Alarm area: Plant

Message = "Relief valve failed to open after 30 seconds"

Condition = PLC1.reliefVlv AND DelayOnTimer

● Logical alarm

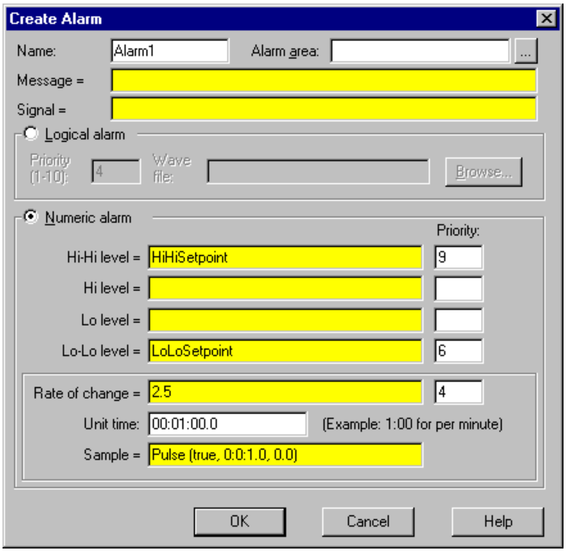
Priority (1-10): 8 Wave file: c:\

○ Numeric alarm

There are two basic alarm types: Logical and Numeric. **When Logical alarm** is selected, Lookout*Direct* prompts you to enter a logical expression in the **Condition** field. It then uses the logical **Condition** to trigger and reset the alarm. If the alarm **Condition** is true, the alarm is active, and if the condition is false, the alarm goes inactive. You can also connect an audio **Wave file** to individual logical alarms. See *Playwave* for additional information.

Lookout*Direct* queues alarm `.wav` files, with up to 100 files in the queue. Each alarm `.wav` file plays completely before the next file plays. If more than 100 alarms fill the queue, new alarms cancel the currently playing files and begin playing instead.

When you pick the **Numeric** alarm selection, the name of the Condition field changes to **Signal**. This prompts you to enter a numeric expression in the field against which your various alarm setpoints are measured. **Hi-Hi**, **Hi**, **Lo**, and **Lo-Lo** are all numeric expressions. **Rate of change** generates an alarm when the signal is actively changing by the set amount for the period of time between any two **Sample** pulses. The **Unit time** setting determines the time units for the rate of change.



Create Alarm

Name: Alarm1
Alarm area:
Message =
Signal =
○ Logical alarm
Priority (1-10): 4
Wave file:
● Numeric alarm

		Priority:
Hi-Hi level =	HiHiSetpoint	9
Hi level =		
Lo level =		
Lo-Lo level =	LoLoSetpoint	6
Rate of change =	2.5	4

Unit time: 00:01:00.0 (Example: 1:00 for per minute)
Sample = Pulse (true, 0:0:1.0, 0.0)

Alarm area specifies the group name associated with the alarm object. All previously defined groups appear in the list box and may be selected with the mouse.

Priority ranges from 1 to 10 where 10 is the most important.

Message is a text expression whose result is displayed in the alarm window when this object generates an alarm. If alarm style is numeric, the relevant alarm trigger prefixes your message (for example, HiHi level: *Alarm message*). See Chapter 9, *Alarms,* in the *LookoutDirect Developer's Manual* for additional information.

Alarm Data Members

Table 2-2. Alarm Data Members

Data Members	Type	Read	Write	Description
active	logical	yes	no	Result of logical alarm status. True if alarm is active and false if alarm is inactive.
hi	logical	yes	no	Result of numeric alarm hi data member status.
hihi	logical	yes	no	Result of numeric alarm hihi data member status.
lo	logical	yes	no	Result of numeric alarm lo data member status.
lolo	logical	yes	no	Result of numeric alarm lolo data member status.
rate	logical	yes	no	Result of numeric alarm rate data member status.

Comments A common alarm condition is caused by a measured value going out of an acceptable range. For example, a storage tank whose level is too low or too high can generate several alarms: `Hihi: Tank level is out of range`, or `Lo: Tank level is out of range`. If you use a numeric style alarm to trigger the alarm and if the tank level fluctuates or “jitters” around the alarm level settings, a new alarm record is generated in the alarm window each time the tank level fluctuates above or below the level settings. To alleviate this condition, you could use the Neutralzone object to filter out minor tank fluctuations.

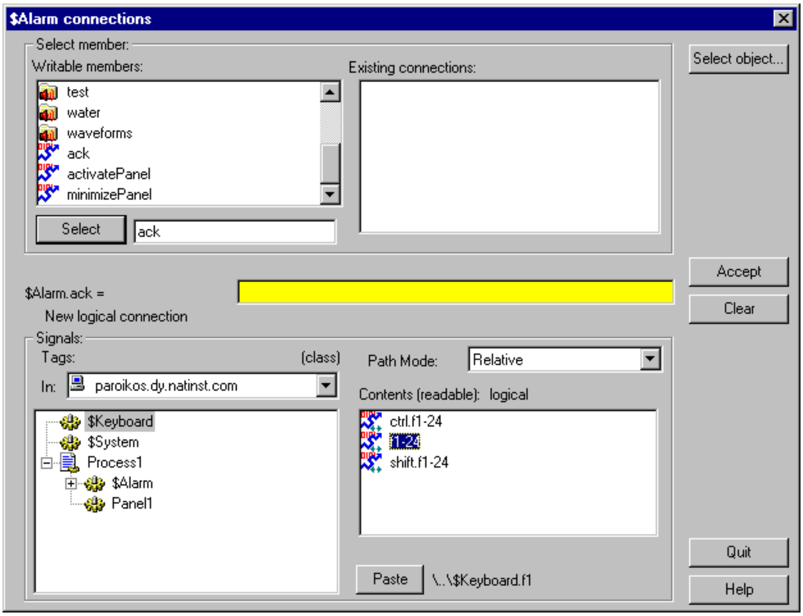
Note Alarms can also be defined through parameter settings on object data members. Many times, this is the most efficient method of defining and creating alarm conditions. See *Database-Generated Alarms* in Chapter 9, *Alarms*, in the *LookoutDirect Developer’s Manual*, and *Editing Object Databases* in Chapter 4, *Using LookoutDirect*, in the *Getting Started with LookoutDirect* manual.

$Alarm

$Alarm is a global object. It makes available global alarm data such as the number of currently active alarms.

You can use $Alarm data members just like other data members. By inserting an expression you can display the number of active alarms in any particular group. Or, by connecting a pushbutton to an acknowledge data member, operators can acknowledge alarms through pushbutton selection.

Assume, for example, that you want to create a pushbutton that acknowledges all alarms. First create a pushbutton object. Next, use the **Object»Edit Connections…** command to connect the pushbutton (`Pb1`) to the `.ack` data member of $Alarm. Such a connection is shown in the following illustration.



$Alarm connections

Select member: ack

$Alarm.ack =

New logical connection

In: paroikos.dy.natinst.com

Path Mode: Relative

Contents (readable): logical

\..\$Keyboard.f1

The expression`(Pb1)`acknowledges all active alarms any time the pushbutton is depressed. You could make similar connections to acknowledge individual alarm areas.

You can define new alarm areas through Alarm objects and by modifying object database parameters. As you create each new alarm area, Lookout*Direct* adds new readable and writable data members to the $Alarm object. $Alarm data members are described in the following table.

$Alarm Data Members

Table 2-3. $Alarm Data Members

Data Member	Type	Read	Write	Description
ack	logical	no	yes	Upon transition from FALSE to TRUE, acknowledges all alarms.
Ackselected	logical	no	yes	Upon transition from FALSE to TRUE, acknowledges all selected alarms.
ActivatePanel	logical	no	yes	Calls Alarm Window upon transition from FALSE to TRUE, making it visible on the screen.
Active	numeric	yes	no	Total number of currently active alarms (that is, alarm conditions that still exist).
Groupname	text	yes	no	Alarm area name associated with the most recent alarm.
Groupname.ack	logical	no	yes	Upon transition from FALSE to TRUE, acknowledges all alarms within the specified group.
Groupname.active	numeric	yes	no	Number of currently active alarms within the specified group.
Groupname.unacked	numeric	yes	no	Number of unacknowledged alarms within the specified group.
Message	text	yes	no	Text of the message of the most recent alarm
MinimizePanel	logical	no	yes	Closes Alarm Window upon transition from FALSE to TRUE, changing it to an icon.
Priority	numeric	yes	no	Alarm priority associated with the most recent alarm.

Table 2-3. $Alarm Data Members (Continued)

Data Member	Type	Read	Write	Description
Priority01.active – Priority10.active	numeric	yes	no	Number of currently active alarms of the specified priority.
Priority01.unacked – Priority10.unacked	numeric	yes	no	Number of unacknowledged alarms of the specified priority.
Silence	logical	no	yes	Turns off alarm sounds when set to TRUE.
Name	text	yes	no	Object name associated with the most recent alarm.
Unacked	numeric	yes	no	Total number of unacknowledged alarms (that is, alarm conditions that have not yet been acknowledged).
Update	logical	yes	no	Pulses high every time there is a new alarm.

Comments Groupname in the preceding table represents the name of any specified alarm area. The $Alarm object contains an `.active`, `.unacked`, and `.ack` data member for every alarm area.

Using $Alarm with Other Objects

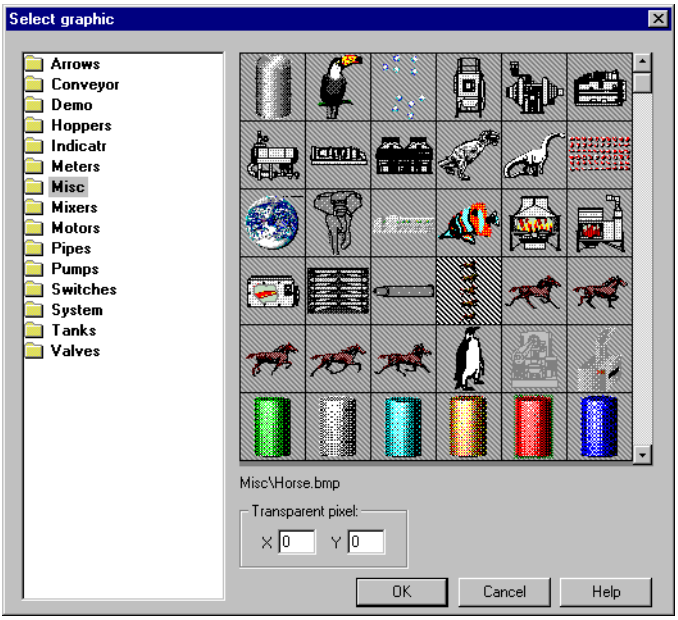
Every time a new alarm appears in Lookout*Direct*, `$Alarm.Message`, `$Alarm.Name`, `$Alarm.Groupname`, and `$Alarm.Priority` are updated with the appropriate information for the new alarm. Then `$Alarm.Update` pulses high to alert you that those four data members contain fresh values.

These data members can serve many purposes, but they are specifically designed to work with the Pager object class. In a typical application, you could use `$Alarm.Update` to initiate a page, possibly including the other four data members in the text of the page. With this system you can also filter which alarms you want to page, and which alarms the pager ignores.

Animator

The Animator object class provides full graphical animation including horizontal and vertical motion, dynamic resizing and visibility, dynamic symbol sequencing and programmable color changes.

When you first create an animator object, the **Select graphic** dialog box appears.



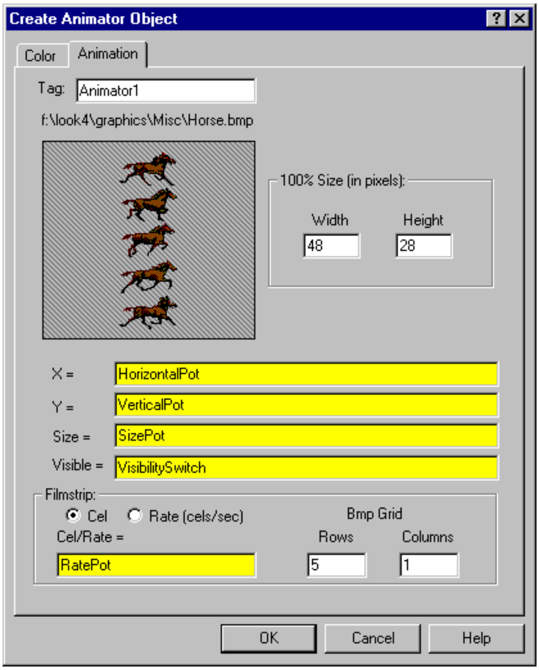
Misc\Horse.bmp

X 0 Y 0

Select the graphic you want to animate. To animate colors, you must select a Windows Metafile (`.wmf`). A moving animation must have the series of images you want to animate arranged in a single bitmap so that each cell of the animation can be defined by a grid of rows and columns.

Animations

To create a moving animation, select the Animation name.



Create Animator Object

Tag: Animator1

f:\look4\graphics\Misc\Horse.bmp

100% Size (in pixels):

Width: 48

Height: 28

X = HorizontalPot

Y = VerticalPot

Size = SizePot

Visible = VisibilitySwitch

Filmstrip: ● Cel ○ Rate (cels/sec)

Cel/Rate = RatePot

Bmp Grid

Rows: 5

Columns: 1

The graphic file you select determines what images appear on the screen. You delineate the images in a bitmap graphic by dividing it into a grid of **Rows** and **Columns**. In the preceding illustration, the single column is `48` pixels **W**ide and each row is `28` pixels **H**igh. Conceivably, the grid *could* consist of 32,000 cells. 100 cells would be more practical.

Each grid cell is a filmstrip image. Internally, the Animator assigns a cell number to each image. It normally plays the filmstrip by progressing from left to right and top to bottom, from the lowest cell number to the highest cell number.

The **Rate (cells/sec)** selection you use to specify a frequency at which the Animator progresses from one cell image to the next. The rate value can be positive or negative and can range from `0.0000005` (one frame every 23.148 days) to `100` frames per second.

If the rate value is negative, the Animator plays the filmstrip backwards, starting at the last cell.

You can use the **Cell** selection in the definition dialog box to identify a particular cell number to display. For example, you might use this selection to display a specific image when a PLC register is equal to a particular value. This is similar to the multistate object, but can provide many more states, depending on the number of cells in the bitmap.

The **X** and **Y** parameters specify the horizontal and vertical position of the image, respectively. These numeric parameters range from `0` to `100`, and together provide the X-Y position of the image as a percent of the Animator dimension on a control panel.

Size is a numeric parameter that ranges from `0` to `100` percent, where `100` percent is the full size of a single cell as specified by **W** and **H**.

Visible is a logical parameter that causes to the image to appear when it goes true and disappear when it goes false.

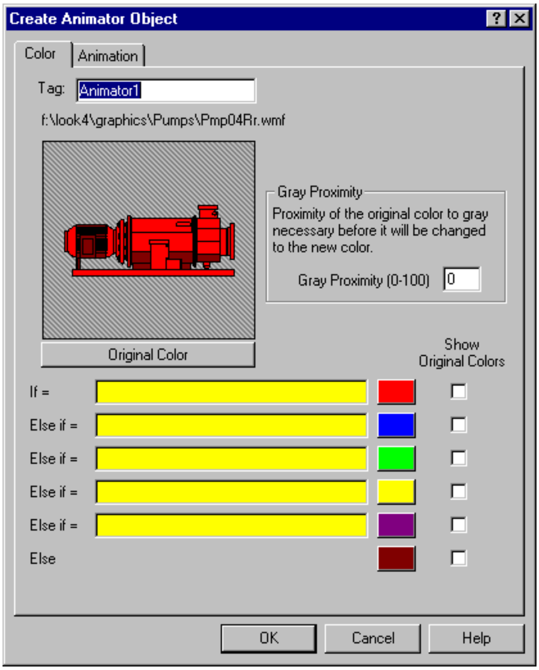
Color Animation

To create color changes programmatically, select the **Color** tab.

With this option, you can set the color map on a `.wmf` graphic using a series of five logical conditions and a default state. If no condition is true, the default state (or color map) is imposed.

You can choose a custom color map by clicking on the color sample associated with each logical condition. You can then select any color from the palette available on your computer.

Select the **Show Original Colors** box if you want the original color map of the `.wmf` file to be imposed for that condition.



Tag: Animator1

f:\look4\graphics\Pumps\Pmp04Rr.wmf

Gray Proximity

Proximity of the original color to gray necessary before it will be changed to the new color.

Gray Proximity (0-100) 0

Gray Proximity determines the color saturation point at which the selected color replaces an original color. When this parameter is set to `0`, most colors change. When this parameter is set to `100`, most colors will not change. You can test the effects of this setting by putting your cursor over the color button and observing the change in the graphic displayed in the dialog box window.

Notice that the if and else if statements are evaluated in sequence. If several conditional expressions are true at once, Multistate displays the graphic associated with the first true expression.

For instance, if your **If** parameters use less-than comparisons, such as `PumpSpeed<50`, the following **Else if** parameter must have a larger comparison value, such as `PumpSpeed<75`. If you use a smaller comparison value, such as `PumpSpeed<25`, the color change will not take place. In other words, the comparison values must be used from smallest to largest.

In the same way, if you are using greater-than comparisons, such as `PumpSpeed>50`, you must list your comparison values from largest to smallest, so that the next **Else If** parameter would have to be something like `PumpSpeed>25`.

Put **If** parameters using equality, such as `PumpSpeed=50`, before parameters using inequalities.

A few minutes experimentation should help you understand the interactions of the color choice conditions.

Animator Data Members

Table 2-4. Animator Data Members

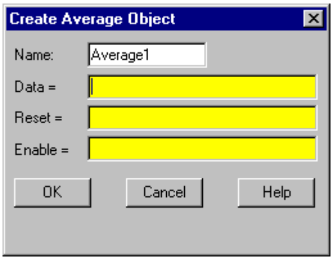
Data Member	Type	Read	Write	Description
none	—	—	—	Animator objects do not have data members

Comments Consider using an Integral object instead of a Counter/Pulse combination when trying to achieve smooth animation.

Related Objects *DialGauge*, *Gauge*, *Multistate*, *Pipe*, *Spinner*

Average

Average actively calculates the average value of **Data** over time. Average is only active when the **Enable** expression is true and resets to zero when the **Reset** expression transitions from off to on. Average also maintains an array of up to 35 previous averaged values. If **Enable** is left blank, the object always actively calculates the average.



Create Average Object

Name: Average1

Data =

Reset =

Enable =

Data is a numeric expression while **Reset** and **Enable** are logical expressions.

Average Data Members

Table 2-5. Average Data Members

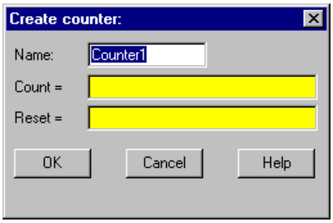
Data Members	Type	Read	Write	Description
(implicit)	numeric	yes	no	Current average calculated since the most recent Reset signal. Updated approximately once per second.
1 – 35	numeric	yes	no	Previous average values. Signal 1 is the most recent prior average since the Reset signal went high.
DataReset	logical	no	yes	Upon transition from FALSE to TRUE, resets to zero all data members—including the current average and all previous averages.

Comments The **Reset** expression could be a regular pulse interval created by a TimeOf*xxxx* timer, so that the pulse is synchronized to the top of the hour or day. For example, if you want to calculate the daily average flow rate, use the output signal from a TimeOfDay timer or a daily Spreadsheet object to reset the average calculation at the beginning of each day. If you want to calculate the average flow rate only when a pump is running, use the input signal from the pump motor relay in the **Enable** parameter.

Related Objects *Minimum*, *Maximum*

Counter

Counter counts the number of times that the **Count** expression transitions from off to on. The digital display shows the number of pulses counted so far, and is updated approximately once per second—however, it can receive and count multiple pulses within a given second. The counter can count to just under 4,503,600,000,000,000 or 142,710 years worth of pulses at one pulse per second. When the **Reset** expression transitions from off to on, the counter resets to zero.



Create counter:

Name: Counter1

Count =

Reset =

Both **Count** and **Reset** are logical expressions.

Counter Data Members

Table 2-6. Counter Data Members

Data Member	Type	Read	Write	Description
(implicit)	numeric	yes	no	numeric total of pulse count

Comments You should not use Counters to count external pulse signals that cycle more often than about once per second. For higher counting speeds, use the accumulator capabilities built into your PLC.

Related Objects *Accumulator*, *Integral*

DataTable

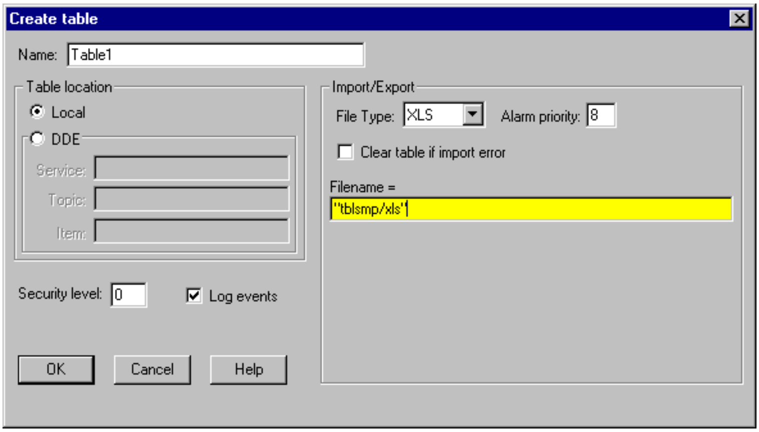
DataTables are used in the following applications:

- Multiplexing various data sources into a single display template.
- Importing or exporting large quantities of data to other applications using DDE. (Unlike the DdeTable, the DataTable provides bidirectional DDE communications. See *DdeTable*.)
- Networking multiple Lookout*Direct* nodes (see Chapter 5, *Networking*, in the *LookoutDirect Developer's Manual*).

A DataTable contains a matrix of rows and columns, much like a spreadsheet. Each column is represented by a letter (A – IV). Each row is represented by a number (1 – 1,000). Letter-number combinations represent intersections of rows and columns. Such intersections are called cells. Any given cell contains a value. A cell value can be numeric, logical, or textual.

Note DataTables are advanced tools that require a mastery of Lookout*Direct* object databases. Make sure you understand editing, connecting to, and using object databases and aliases before creating a DataTable object. See Chapter 4, *Using LookoutDirect*, and Chapter 5, *Developer Tour*, in the *Getting Started with LookoutDirect* manual for more information on object databases.

The following dialog box appears when you create a DataTable:



Create table

Name: Table1

Table location

● Local

○ DDE

Service:

Topic:

Item:

Import/Export

File Type: XLS

Alarm priority: 8

□ Clear table if import error

Filename = "tblsmp/xls"

Security level: 0

☑ Log events

Specify **DDE** parameters if you want to use the DataTable to exchange large quantities of data from an external source (such as Excel) continuously, with both Lookout*Direct* and the other program running.

See Chapter 6, *Dynamic Data Exchange*, of the *LookoutDirect Developer’s Manual* for more information on DDE connections to other programs.

Specify **Local** if you are using the DataTable to import or export data from Lookout*Direct* into or out of a static file. Also specify **Local** if you are using the DataTable to multiplex various signals into a single display or graphic template.

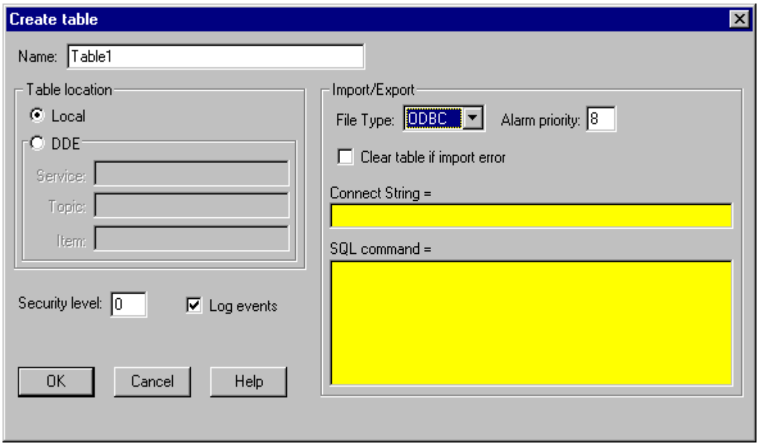
To exchange data with another DataTable running in another Lookout*Direct* process, you can use ordinary Lookout*Direct* connections.

The export and import data members control the transfer of data between Lookout*Direct* and a Microsoft Excel spreadsheet. The spreadsheet filename must be entered in the **Import/Export** section of the **Create table** dialog box.

Note The data table import and export data members only work with Microsoft Excel Version 4 at this time. This feature does not work with Excel Version 5 or greater.

Instead of entering a file name in the **Import/Export** dialog box, you can enter a Lookout*Direct* expression in the **Filename** field. You can then import and export different files using a switch setting, a text entry box, or some other expression input. If something goes wrong with the transfer of data, including data corruption, Lookout*Direct* reports an alarm.

Click on the **Clear table if import error** checkbox to clear the DataTable before importing either from a spreadsheet or an ODBC database. This only affects what happens if there is an error during the import. If there are no errors during the import, the imported data always replaces the old data in the table. However, if the checkbox is selected, the DataTable is always cleared before starting an import, so that, if an error occurs, the DataTable is left empty. If the checkbox is not checked and an error occurs, the old data is left in the table.



Create table

Name: Table1

Table location

● Local

○ DDE

Service:

Topic:

Item:

Security level: 0

☑ Log events

Import/Export

File Type: ODBC

Alarm priority: 8

□ Clear table if import error

Connect String =

SQL command =

The **Connect String=** field specifies the DataSource Name (DSN) as well as any other parameters needed by the ODBC driver to make the connection to your chosen data source.

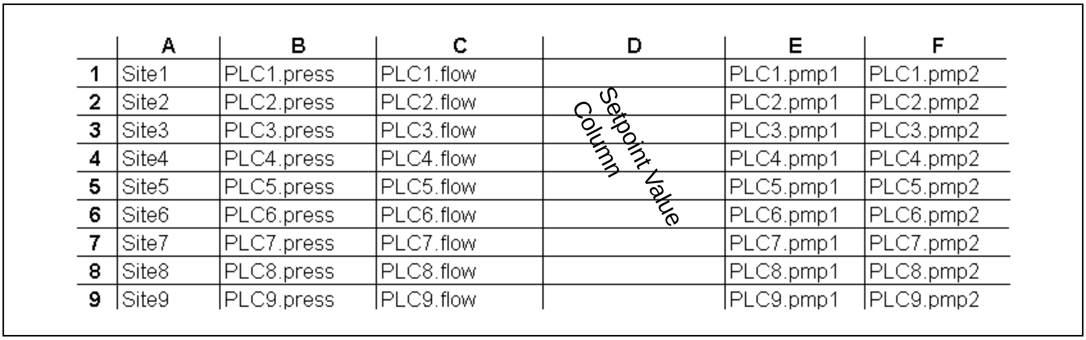
The **SQL command=** field is where you enter the SQL string you want to pass to the ODBC driver.

Use the **Clear table if import error** checkbox to clear the DataTable before importing either from an ODBC database or a spreadsheet. This only affects what happens if there is an error during the import. If there are no errors during the import, the imported data always replaces the old data in the table. However, if the checkbox is selected, the DataTable is always cleared before starting an import, so that, if an error occurs, the DataTables left empty. If the checkbox is not checked and an error occurs, the old data is left in the table.

Multiplexing Displays and Graphics

You can use DataTables to multiplex signals into a single control panel, used as a template. For instance, assume you have nine identical pump stations. At each site you have a single PLC monitoring the pressure, flow rate, and status of two pumps. You are also controlling an analog output with an operator setpoint from Lookout*Direct*. Instead of developing nine identical control panels in Lookout*Direct*, you can build just one panel and multiplex the signals from the nine sites into a single set of graphics.

The following figure is a graphical representation of the connections for a possible DataTable.



	A	B	C	D	E	F
1	Site1	PLC1.press	PLC1.flow		PLC1.pmp1	PLC1.pmp2
2	Site2	PLC2.press	PLC2.flow		PLC2.pmp1	PLC2.pmp2
3	Site3	PLC3.press	PLC3.flow		PLC3.pmp1	PLC3.pmp2
4	Site4	PLC4.press	PLC4.flow		PLC4.pmp1	PLC4.pmp2
5	Site5	PLC5.press	PLC5.flow		PLC5.pmp1	PLC5.pmp2
6	Site6	PLC6.press	PLC6.flow		PLC6.pmp1	PLC6.pmp2
7	Site7	PLC7.press	PLC7.flow		PLC7.pmp1	PLC7.pmp2
8	Site8	PLC8.press	PLC8.flow		PLC8.pmp1	PLC8.pmp2
9	Site9	PLC9.press	PLC9.flow		PLC9.pmp1	PLC9.pmp2

Note At this time there is no actual view in Lookout*Direct* for you to see this table. It is a representation, not a screen capture of a Lookout*Direct* dialog box or display element.

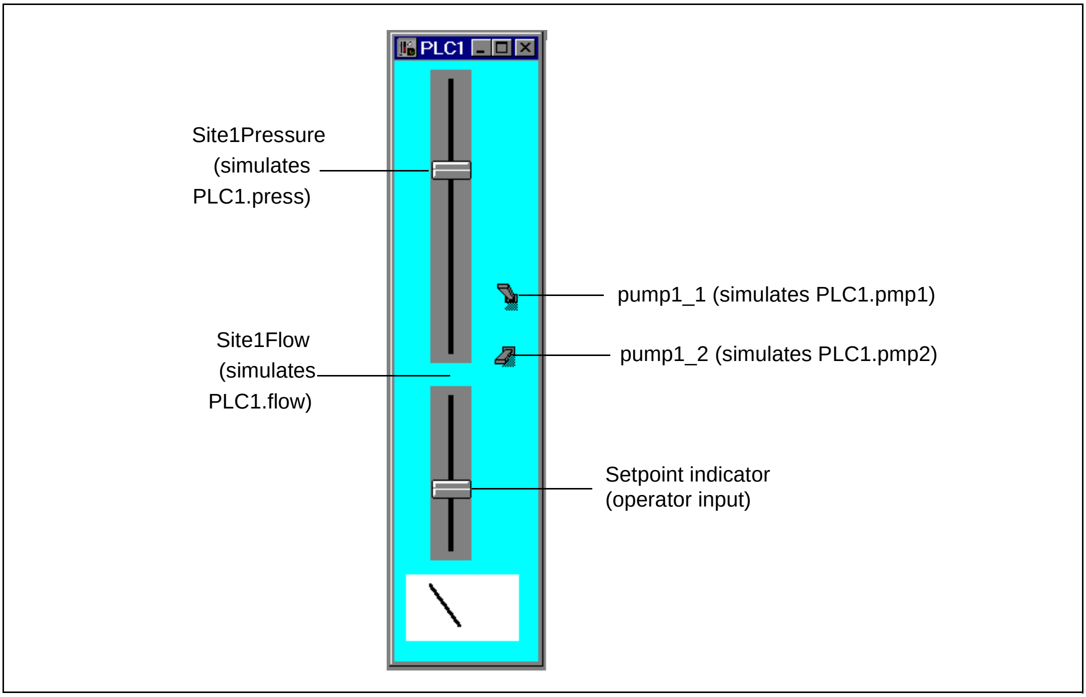
Each row of the table is connected to the site name, the data you want to keep track of from the PLC at that site, and a place for the operator setpoint data for that site.

Notice the names of the nine PLCs (`PLC1, PLC2`…). Each value has an alias (`press`, `flow`, `pmp1`, `pmp2`). Each cell represents a connection made from that PLC output to the that cell of the DataTable.

The open column D represents the connection of a single Pot object called `Setpoint` to the column—not to individual cells. This way Lookout*Direct* only changes the value of a cell in column D when that particular row has been made active. You connect the individual cells in column D (D1 through D9) to the correct holding registers (outputs) on the respective PLCs. For example, `Table1.D1` would be connected to the appropriate data member of `PLC1 — Table1.D2` to the appropriate data member of `PLC2`, and so on. In other words, when you intend to multiplex signals to a panel through a data table connect inputs from a PLC or RTU to individual cells. Connect operator setpoints (outputs) to columns.

DataTable Example

To practice using a DataTable, you can create a PLC simulator as shown in the following illustration. You can use a separate control panel for each PLC output simulator, or place all the objects you make on one panel.


PLC1
Site1Pressure (simulates PLC1.press)
pump1_1 (simulates PLC1.pmp1)
Site1Flow (simulates PLC1.flow)
pump1_2 (simulates PLC1.pmp2)
Setpoint indicator (operator input)

Use two slider pots as your pressure and flow outputs, and two switches as the on/off indicators for pumps one and two. You can use the dial gauge at the bottom to check the operator input, `Setpoint`. Instead of connecting the cells of your data table to PLC outputs, you connect directly to the Lookout*Direct* objects, named after the outputs for clarity.

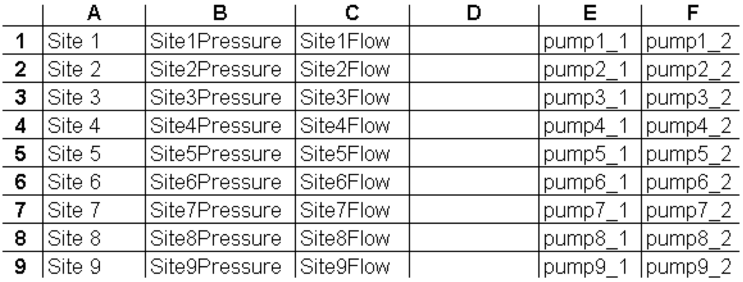
You should create at least three sets of PLC input/output simulators. The following example refers to the 9 PLCs mentioned in the *Multiplexing Displays and Graphics* section, but you can explore using a data table using 2 or 3 simulators.

Connect specific DataTable cells from the `D` column to the Setpoint indicator for each PLC simulator.

After you have built your simulators, open a new Control Panel to use as your display and create a DataTable. You will create the rest of the display in a later step.

When it first appears, the DataTable contains the number 0. This indicates the contents of cell A1. You can increase the size of the display window, but you cannot show the entire array of data in the table. You can view the contents of any cell in column A by clicking on the window when you are in run mode. Selecting the contents of that cell activates that row of the data table.

Enter text expressions into the cells of column A to act as your table index. For example, enter the string `"Site 1"` as the connection to your DataTable `A1.txt` data member. Connect the outputs of your PLC simulators to the cells in the B, C, E, and F columns shown in the following figure. You will connect an operator input to the entire column D, and than later connect individual cells in the D column to your PLC simulators.



	A	B	C	D	E	F
1	Site 1	Site1Pressure	Site1Flow		pump1_1	pump1_2
2	Site 2	Site2Pressure	Site2Flow		pump2_1	pump2_2
3	Site 3	Site3Pressure	Site3Flow		pump3_1	pump3_2
4	Site 4	Site4Pressure	Site4Flow		pump4_1	pump4_2
5	Site 5	Site5Pressure	Site5Flow		pump5_1	pump5_2
6	Site 6	Site6Pressure	Site6Flow		pump6_1	pump6_2
7	Site 7	Site7Pressure	Site7Flow		pump7_1	pump7_2
8	Site 8	Site8Pressure	Site8Flow		pump8_1	pump8_2
9	Site 9	Site9Pressure	Site9Flow		pump9_1	pump9_2

Connecting Signals to DataTables

To connect a value to a particular cell or column within a DataTable, use the **Object»Edit Connections…** command. Select the specific data member of the DataTable to be written, and identify the data member of the source object.

Connecting to Cells

The value from `Site1Pressure` is numeric. To write the value of `Site1Pressure` into cell `B1`, connect the simulated output to `B1`—not `B1.logical` or `B1.txt`. The **Edit connection** dialog box is shown in the following illustration.

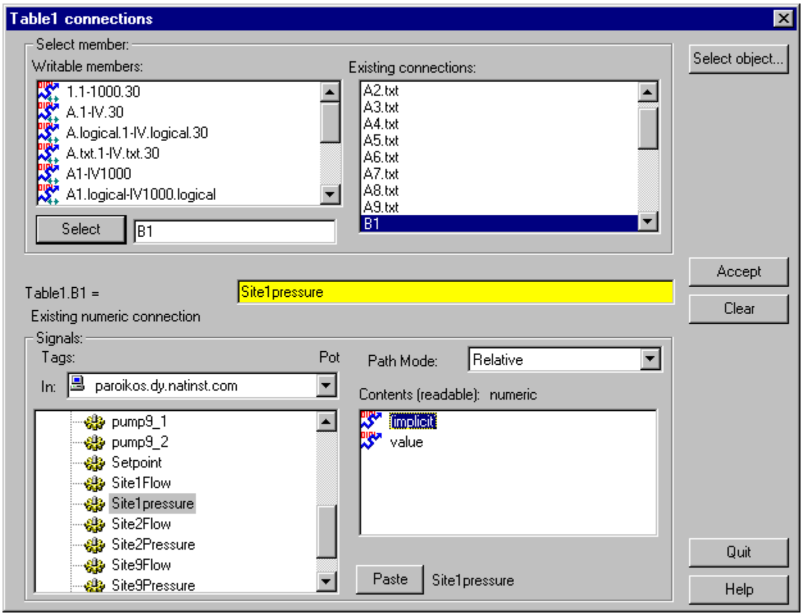


Table1 connections

Select member:

Writable members:

- 1.1-1000.30
- A.1-IV.30
- A.logical.1-IV.logical.30
- A.txt.1-IV.txt.30
- A1-IV1000
- A1.logical-IV1000.logical

Select: B1

Existing connections:

- A2.txt
- A3.txt
- A4.txt
- A5.txt
- A6.txt
- A7.txt
- A8.txt
- A9.txt
- B1

Table1.B1 = Site1pressure

Existing numeric connection

Signals:

Tags: In: paroikos.dy.natinst.com

- pump9_1
- pump9_2
- Setpoint
- Site1Flow
- Site1pressure
- Site2Flow
- Site2Pressure
- Site9Flow
- Site9Pressure

Path Mode: Relative

Contents (readable): numeric

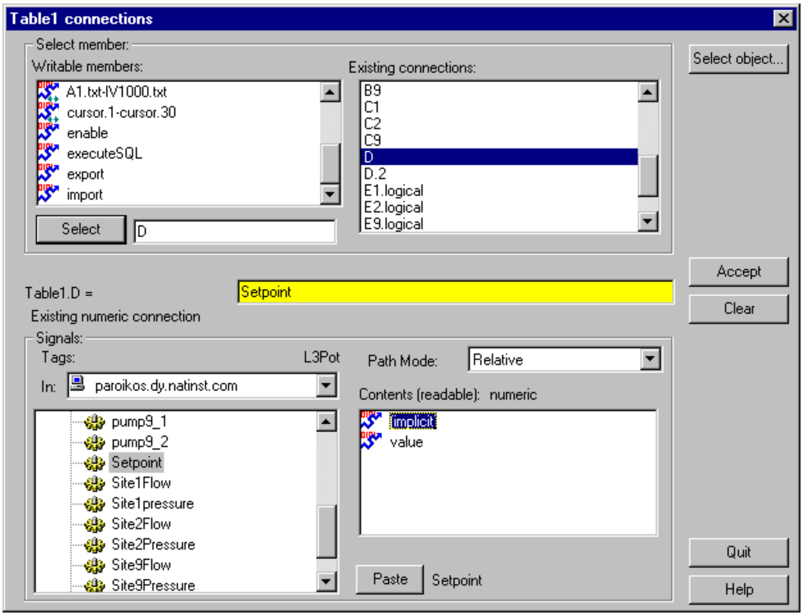
- implicit
- value

Paste: Site1pressure

After you establish a connection to a cell, the value within the cell changes any time the expression changes. Any time the value of `Site1Pressure` changes, the value within cell `B1` changes.

Connecting to Columns

To connect the value of the `Setpoint` potentiometer to column D, select `D`—not `D.logical` or `D.txt`—as shown in the following illustration. As with the previous value, you use `D` because you are using a numerical value. There is no number following the `D` because you are connecting to a column, not a cell.



Select: D

Table1.D = Setpoint

Existing numeric connection

Tags: L3Pot

In: paroikos.dy.natinst.com

Path Mode: Relative

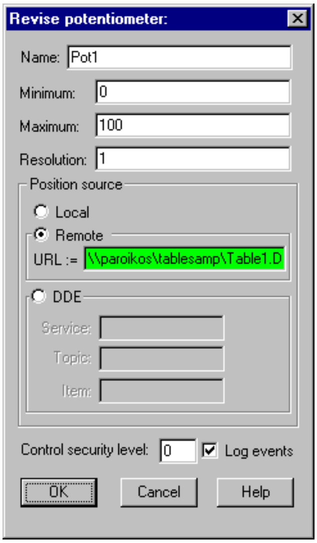
Contents (readable): numeric

Paste Setpoint

Once a connection to a column is established, the value within the cell *at the currently selected row* changes when the expression changes. So, if you activate row 4 and change the value of `Setpoint` changes, the value within cell `D4` changes. No other cells are affected until you move the cursor to activate another row and the operator adjusts the pot.

Reading a Cell Value Back to a Lookout*Direct* Object

When you connect the display panel potentiometer called `Setpoint` to `Table1.D`, you should configure your pot with the appropriate **Remote** parameter to ensure that it automatically adjusts to track the value in any cell (within the specified column) when the cursor moves to a new row.



Name: Pot1

Minimum: 0

Maximum: 100

Resolution: 1

Position source

○ Local

● Remote

URL := \\paroikos\tablesamp\Table1.D

○ DDE

Service:

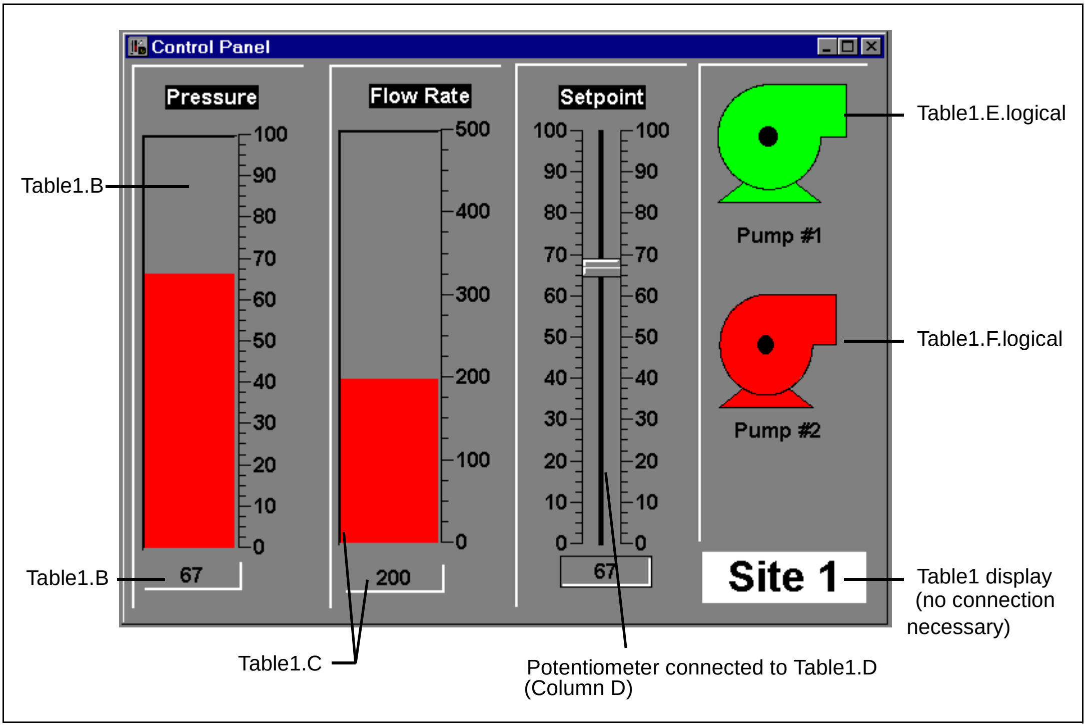
Topic:

Item:

Control security level: 0 ☑ Log events

The Display Panel

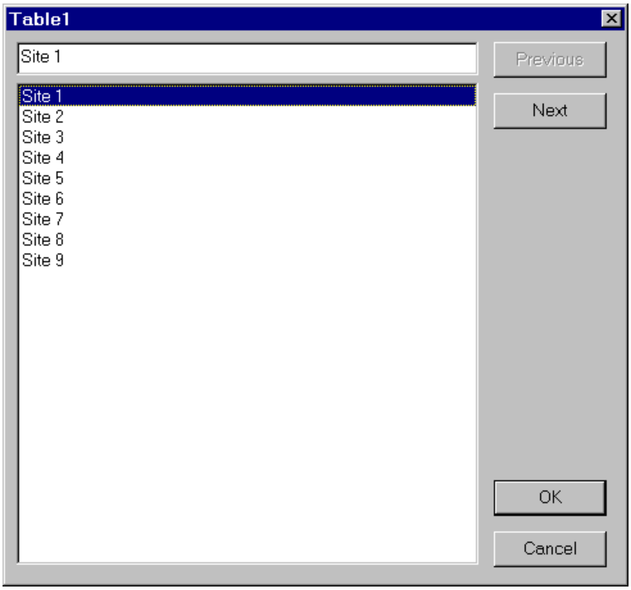
Now that your DataTable (called `Table1`) is complete, you can return to your display panel (where you placed the DataTable object initially) and begin to build a single control panel that allows you to multiplex the data from your PLCs. A sample panel is shown in the following illustration.



Instead of using expressions that reference the actual values from your driver objects to display values, use the column names from `Table1`. For instance, the bar graph and numeric readout for Pressure both represent the expression `Table1.B`. The actual value for `Table1.B` depends on the what row is currently active in the DataTable. In the illustration row 1 is active, so all the expressions return the value in their respective columns at row 4. The callouts in the picture indicate how the control panel was developed. All are Lookout*Direct* expressions except the Pot and the DataTable, which are displayable objects.

Operating Your Multiplexed Panel

The plate in bottom right corner of the control panel is the DataTable object (`Table1`). To view a different site with the control panel (that is, to activate a different row, thereby selecting a different PLC), click on `Table1` and the following dialog box should appear.



This list box gets the row names from Column A in the Data Table

By selecting a site, you connect your control panel to the PLC at that site. This is referred to as moving the cursor to that row.

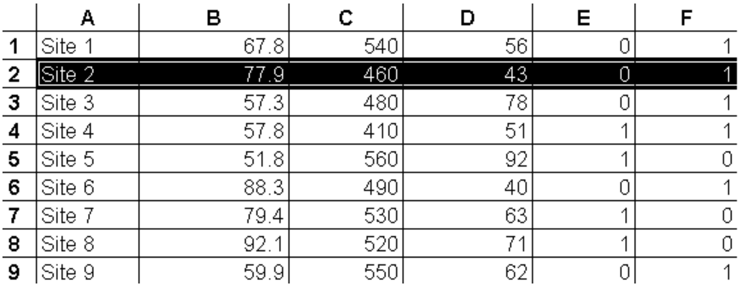
DataTable Cursors

The cursor is a DataTable pointer that you can move from row to row to activate the values in the cells of that row. There are several methods for controlling the location of the cursor:

- Connect a numeric expression to the `cursor` data member. A typical example would be to connect a potentiometer (minimum = 1, maximum = the number of rows in table, resolution = 1) to the cursor.
- Connect logical expressions to appropriate row numbers. A typical example would be to create a pushbutton for every row and then connect them to their respective row numbers.

- Use the display (list box) built into the DataTable object. A typical example was described previously where you connected text values to cells in column A and then displayed the table as a plate.

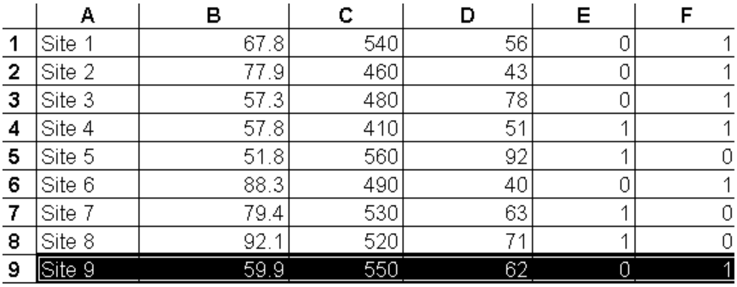
The following example provides a graphical representation of a DataTable (called `Table1`) showing typical values within its many cells. You can create a multiplex effect in Lookout*Direct* by referencing column names and then selecting the row with the information you want to use. If you reference column names (instead of individual cells), the DataTable outputs only the values within the currently selected row. If you reference individual cells, the DataTable outputs the current value within the cell—no matter where the cursor is.



	A	B	C	D	E	F
1	Site 1	67.8	540	56	0	1
2	Site 2	77.9	460	43	0	1
3	Site 3	57.3	480	78	0	1
4	Site 4	57.8	410	51	1	1
5	Site 5	51.8	560	92	1	0
6	Site 6	88.3	490	40	0	1
7	Site 7	79.4	530	63	1	0
8	Site 8	92.1	520	71	1	0
9	Site 9	59.9	550	62	0	1

Outputs =	**Site2**	**77.9**	**460**	**78**	**0**	**1**

The value of cell B2 is currently 77.9. Therefore, the `Table1.B` data member is also 77.9. If you move the cursor to row 9, the value of `Table1.B` changes to 59.9, as shown.



	A	B	C	D	E	F
1	Site 1	67.8	540	56	0	1
2	Site 2	77.9	460	43	0	1
3	Site 3	57.3	480	78	0	1
4	Site 4	57.8	410	51	1	1
5	Site 5	51.8	560	92	1	0
6	Site 6	88.3	490	40	0	1
7	Site 7	79.4	530	63	1	0
8	Site 8	92.1	520	71	1	0
9	Site 9	59.9	550	62	0	1

Outputs =	**Site9**	**59.9**	**550**	**62**	**0**	**1**

Using Multiple Cursors

The previous description assumes you are using just one cursor. But a DataTable can have up to 20 cursors. Multiple cursors allow you to select multiple rows at the same time. When using multiple cursors, you also use multiple names for each column. For a given column, each name is associated with a given cursor. If you are using two cursors in the previous example (`cursor` and `cursor.2`), you can reference the column name of a given column as follows:

Table 2-7. Column Names

	A	**B**	**C**	**D**	**E**	**F**
cursor	A.txt	B	C	D	E.logical	F.logical
cursor.2	A.txt.2	B.2	C.2	D.2	E.logical.2	F.logical.2

Earlier, when you were using just one cursor, you connected the value of a potentiometer called `Setpoint` to column D. Subsequently the value of `Setpoint` was written to the cell at the row selected by the cursor. But when there are multiple cursors, you have to select which cursor you are writing to. Thus, depending on how you want your table to work, you might connect the potentiometer to both `Table1.D` and `Table1.D.2`.

DataTable Data Members

Table 2-8. DataTable Data Members

Data Members	Type	Read	Write	Description
executeSQL	logical	no	yes	Imports data from an ODBC database into the DataTable when value goes high, using the connect string and SQL command you enter in the Create/modify DataTable dialog box.
(implicit)	DdeTable	no	no	Not displayable in Lookout*Direct*, but it can be referenced by a DDE link from another application.

Table 2-8. DataTable Data Members (Continued)

Data Members	Type	Read	Write	Description
1.1–1000.20	logical	no	yes	Specifies row (1, 2, 3, …1000) or specifies row.cursor (for example, 24.2 is the selector for row 24, cursor 2). Upon transition from false to true, the specified cursor moves to specified row.
A.1–IV.20	numeric	yes	yes	Specifies column names (for example, A, B, C…IV) or specifies column names and associated cursor numbers (such as, A.1, B.1, A.2, B.2, and so on.) *Read*—returns a numeric value from the cell specified by the column and currently selected row of the indicated cursor. *Write*—writes a numeric value into the cell specified by the column and currently selected row of the indicated cursor.
A.logical.1 – IV.logical.20	logical	yes	yes	Specifies column names (for example, A.logical, B.logical, C.logical, …IV.logical) or specifies column names and associated cursor numbers (such as, A.logical.1, B.logical.1, A.logical.2, B.logical.2, and so on.) *Read*—returns a logical value from the cell specified by the column and currently selected row of the indicated cursor. *Write*—writes a logical value into the cell specified by the column and currently selected row of the indicated cursor.

Table 2-8. DataTable Data Members (Continued)

Data Members	Type	Read	Write	Description
A.txt.1 – IV.txt.20	text	yes	yes	Specifies column names (for example, A.txt, B. txt, C. txt, …IV. txt) or specifies column names and associated cursor numbers (such as, A. txt.1, B. txt.1, A. txt.2, B. txt.2, and so on.) *Read*—returns a text value from the cell specified by the column and currently selected row of the indicated cursor. *Write*—writes a text value into the cell specified by the column and currently selected row of the indicated cursor.
A1 – IV16384	numeric	yes	yes	Specified cell interpreted as numeric value
A1.logical –IV16384.logical	logical	yes	yes	Specified cell interpreted as logical value
A1.txt –IV16384.txt	text	yes	yes	Specified cell interpreted as text value
Cursor.1 – Cursor.20	numeric	yes	yes	Specifies the currently selected row of the indicated cursor.
enable	logical	yes	yes	If TRUE (the default), enables DDE. If FALSE, disables DDE. The default value is on. The input is ignored for non-DDE TextEntry objects.
export	logical	no	yes	When this input transitions from false to true, the Lookout*Direct* data table is exported to the designated spreadsheet file.
import	logical	no	yes	When this input transitions from false to true, the Lookout*Direct* data table imports data from the designated spreadsheet.

Table 2-8. DataTable Data Members (Continued)

Data Members	Type	Read	Write	Description
Modified	logical	yes	no	Pulses TRUE when any cell value in the DataTable changes.
Update	logical	yes	no	Pulses each time the cursor changes rows. Often used to “call up” a control panel.

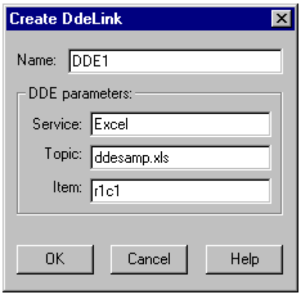
Related Objects *DdeTable*, *DdeLink*

DdeLink

DdeLink creates a unidirectional Dynamic Data Exchange (DDE) link to another application. The other application could be running on the same computer or on another computer over a network. DdeLink objects provide an easy way to *import* remote values into Lookout*Direct*. See Chapter 5, *Dynamic Data Exchange*, in the *LookoutDirect Developer's Manual* for more information on DDE. For each DdeLink object you define, Lookout*Direct* creates a separate link to the other application. If you need to import large quantities of data, you should use the DdeTable or DataTable object. See *DdeTable* and *DataTable* for more information.

DdeLinks on Same Computer

If you are importing values from another application running on the same computer, your DDE parameters will look similar to the ones in the following illustration.



Create DdeLink

Name: DDE1

DDE parameters:

Service: Excel

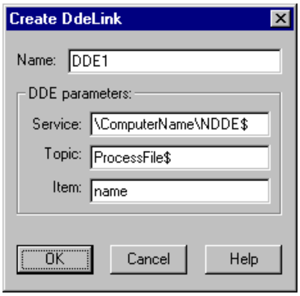
Topic: ddesamp.xls

Item: r1c1

Service specifies the application name, **Topic** specifies the file, and **Item** points to the individual value (r1c1 refers to the cell at row1, column1. Unfortunately, Excel does not support the more convenient A1 cell references with DDE).

DdeLinks to Remote Computer

If you are importing values from another application running on a remote computer, your DDE parameters will look similar to the ones in the following illustration.



Name: DDE1

Service: \ComputerName\NDDE$

Topic: ProcessFile$

Item: name

Notice the differences in the **Service**, **Topic**, and **Item** parameters. The backslashes (`\\`) and dollar signs (`$`) are standard requirements for making network connections in Microsoft Windows. `ComputerName` specifies the network name of the computer you are connecting to. If you are connecting to a value in another Lookout*Direct* application, `ProcessFile` is the Lookout*Direct* file name running on the remote computer, and `Name` refers to the name you are linking to.

DdeLink Data Members

Table 2-9. DdeLink Data Members

Data Members	Type	Read	Write	Description
(implicit)	numeric	yes	no	DDE link interpreted as numeric value.
enable	logical	yes	yes	If TRUE (the default), enables DDE. If FALSE, disables DDE. This input is ignored for non-DDE TextEntry objects.
hot	logical	yes	no	Status of DDE link.
logical	logical	yes	no	DDE link interpreted as logical value.
txt	text	yes	no	DDE link interpreted as text value.

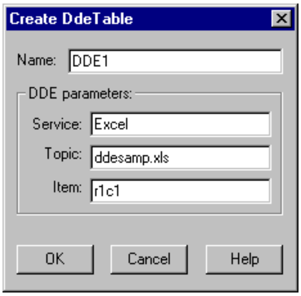
Related Objects *DdeTable*, *DataTable*

DdeTable

DdeTable creates a unidirectional Dynamic Data Exchange (DDE) link to another application. The other application could be running on the same computer or on another computer over a network. See Chapter 5, *Dynamic Data Exchange*, in the *LookoutDirect Developer's Manual* for more information on DDE. You can use DdeTable to *import* large quantities of data from other applications. The table format is much more efficient at transferring data than the Link format because a table can contain hundreds or even thousands of data points that all share a single link. On the other hand, the link format can only transfer a single value per link—and every link requires a certain amount of CPU overhead. If you are only importing a relatively small amount of data, you may find the DdeLink technique easier to implement.

DdeTable on Same Computer

If you are importing values from another application running on the same computer, your DDE parameters will look similar to the ones in the following figure.



Create DdeTable

Name: DDE1

DDE parameters:

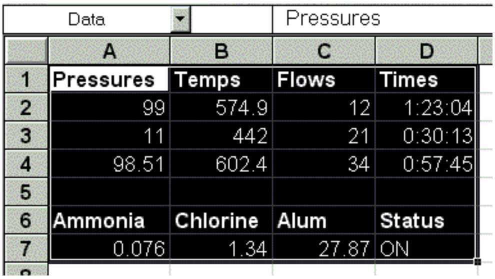
Service: Excel

Topic: ddesamp.xls

Item: r1c1

Service specifies the application name, **Topic** typically specifies the file, and **Item** specifies the particular data item name.

The following example shows an Excel spreadsheet with the highlighted range named Data.

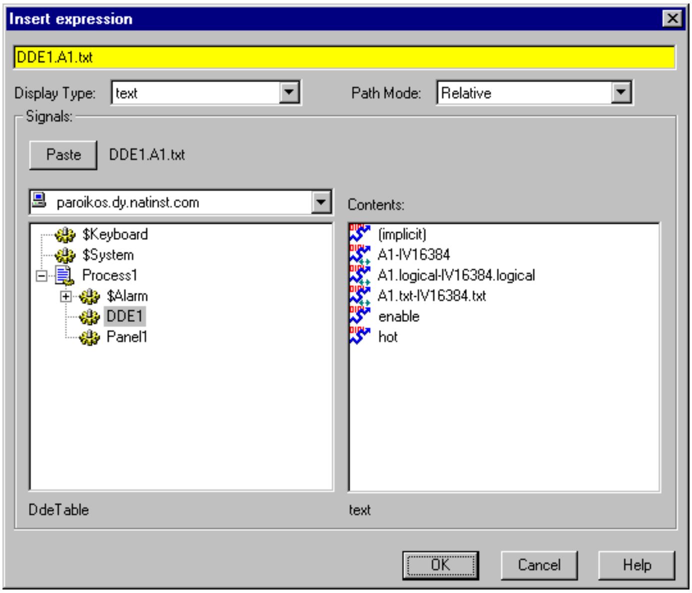


Data | Pressures

	A	B	C	D
1	Pressures	Temps	Flows	Times
2	99	574.9	12	1:23:04
3	11	442	21	0:30:13
4	98.51	602.4	34	0:57:45
5				
6	Ammonia	Chlorine	Alum	Status
7	0.076	1.34	27.87	ON

You can now display any value in the range Data with this Lookout*Direct* DdeTable object.

Select **Insert»Expression...** and then choose the DdeTable data member corresponding to the Excel spreadsheet cell containing the value you want to display. Make sure that the type of data member you select matches the type data in the spreadsheet cell.



Insert expression

DDE1.A1.txt

Display Type: text

Path Mode: Relative

Signals:

Paste DDE1.A1.txt

paroikos.dy.natinst.com

- $Keyboard
- $System
- Process1
  - $Alarm
  - DDE1
  - Panel1

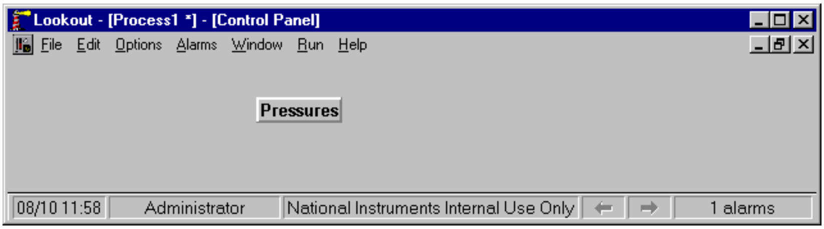
Contents:

- (implicit)
- A1-IV16384
- A1.logical-IV16384.logical
- A1.txt-IV16384.txt
- enable
- hot

DdeTable

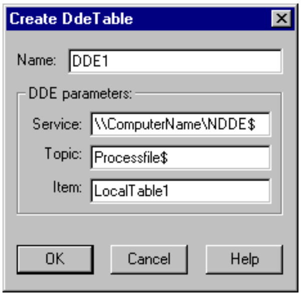
text

Notice the reference for the displayed item in the Lookout*Direct* status bar.


Lookout - [Process1 *] - [Control Panel]
File
Edit
Options
Alarms
Window
Run
Help
Pressures
08/10 11:58
Administrator
National Instruments Internal Use Only
1 alarms

DdeTable to Remote Computer

Using the DdeTable object over a network is somewhat different from the previous example.



Name: DDE1

Service: \\ComputerName\NDDE$

Topic: Processfile$

Item: LocalTable1

Notice the differences in the **Service**, **Topic**, and **Item** parameters. The backslashes (\\) and dollar signs ($) are standard requirements for making network connections in Windows. `ComputerName` specifies the network name of the computer you are connecting to. If you are connecting to a DdeTable or DataTable in another Lookout*Direct* application, `ProcessFile` is the Lookout*Direct* file name running on the remote computer, and `LocalTable1` refers to the DdeTable or DataTable object you are linking to.

DDETable Data Members

Table 2-10. DdeTable Data Members

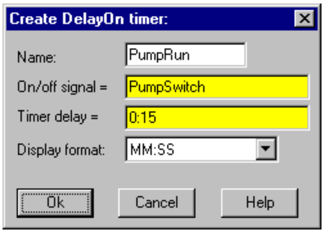
Data Members	Type	Read	Write	Description
(implicit)	numeric	yes	no	Cell A1 interpreted as a numeric value
A1 – IV16384	numeric	yes	no	Specified cell interpreted as a numeric value
A1.logical – IV16384.logical	logical	yes	no	Specified cell interpreted as a logical value
A1.txt – IV16384.txt	text	yes	no	Specified cell interpreted as a text value
enable	logical	yes	yes	If TRUE (the default), enables DDE. If FALSE, disables DDE. This input is ignored for non-DDE TextEntry objects.
hot	logical	yes	no	Status of DDE link

Related Objects *DataTable, DdeLink*

DelayOff

DelayOff is an adjustable delay timer. When **On/off signal** transitions to off, the **Timer delay** begins to count down. At the end of the delay countdown, the output signal turns off. **On/off signal** must remain off during the time delay period for the output signal to turn off. The output immediately turns on when the **On/off signal** turns on.

Timer delay can range from 0.0 seconds to several years, and the effective resolution is 0.1 seconds over the entire range. The timer display digitally shows the time delay remaining, and is updated approximately once per second. If the **On/off signal** is high, the timer display shows `on`. If the **Timer delay** period has expired, the display shows `off`.



Create DelayOn timer:

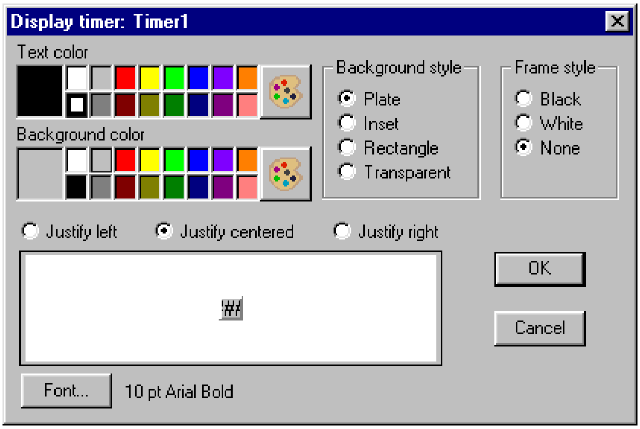
Name: PumpRun

On/off signal = PumpSwitch

Timer delay = 0:15

Display format: MM:SS

The **On/off signal** is a logical expression while **Timer delay** is a numeric expression. Normally, this is a simple time constant such as 0:20—twenty seconds. See *Numeric Data Members* in Chapter 2, *Introduction*, of the *Getting Started with LookoutDirect* manual for information on entering time constants.


Display timer: Timer1

Text color

Background color

Background style
● Plate
○ Inset
○ Rectangle
○ Transparent

Frame style
○ Black
○ White
● None

○ Justify left ● Justify centered ○ Justify right

Font... 10 pt Arial Bold

DelayOff Data Members

Table 2-11. DelayOff Data Members

Data Member	Type	Read	Write	Description
(implicit)	logical	yes	no	Logical timer value

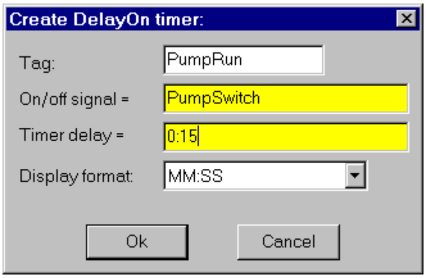
Comments The DelayOff timer can prevent a pump from short-cycling.

Related Objects *DelayOn*, *Interval*, *OneShot*, *Pot*, *TextEntry*

DelayOn

DelayOn is an adjustable delay timer. When **On/off signal** transitions to on, the **Timer delay** begins to count down. At the end of the delay countdown, the output signal turns on. **On/off signal** must remain on during the time delay period for the output signal to turn on. The output immediately turns off when the **On/off signal** turns off.

Timer delay can range from 0.0 seconds to several years, and the effective resolution is 0.1 seconds over the entire range. The timer display digitally shows the time delay remaining and is updated approximately once per second. The timer display shows `off` when the **On/off signal** is low. If the **Timer delay** period has expired, the display shows `on`.



Create DelayOn timer:

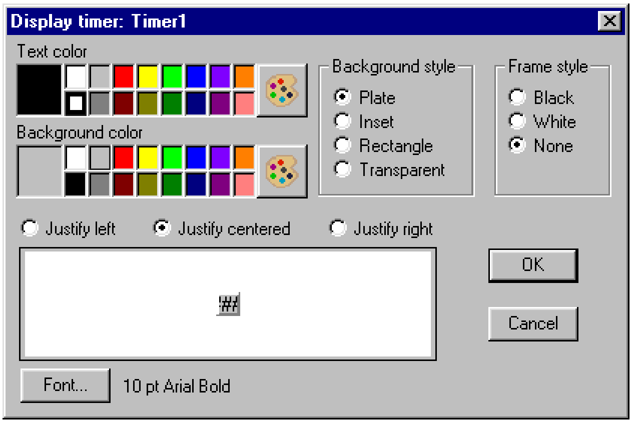
Tag: PumpRun

On/off signal = PumpSwitch

Timer delay = 0:15

Display format: MM:SS

The **On/off signal** is a logical expression while **Timer delay** is a numeric expression. Normally, this is a simple time constant such as 0:20—twenty seconds. See *Numeric Data Members* in Chapter 2, *Introduction*, of the *Getting Started with LookoutDirect* manual for information on entering time constants.



Display timer: Timer1

Text color

Background color

Background style
● Plate
○ Inset
○ Rectangle
○ Transparent

Frame style
○ Black
○ White
● None

○ Justify left ● Justify centered ○ Justify right

10 pt Arial Bold

DelayOn Data Members

Table 2-12. DelayOn Data Members

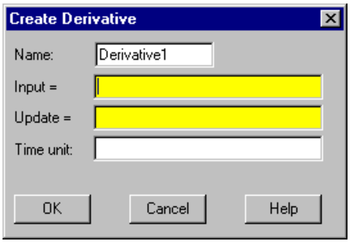
Data Member	Type	Read	Write	Description
(implicit)	logical	yes	no	Logical timer value

Comments The DelayOn timer can be used to prevent pumps from cycling too often, to allow one operation to complete before another begins, or to require a condition to exist for a period of time before an alarm is activated.

Related Objects *DelayOff*, *Interval*, *OneShot*, *Pot*, *Text Entry*

Derivative

Derivative can also be called a rate of change object—it calculates the rate of change of the incoming numeric input signal. You can use this class to calculate the rate at which a tank is filling or draining, or to convert a changing totalized flow value into a flow rate. The output units are in Input Units/Time Unit.



Create Derivative

Name: Derivative1

Input =

Update =

Time unit:

The previous example calculates the rate of change in the water level of a tank. Lookout*Direct* polls the RTU connected to the tank level transmitter every 5 minutes, so this example uses the RTU update data member as the Update pulse for the Derivative object. Tanklevel is in feet and Time unit is 1 minute, so the output result is in feet per minute.

Input is the numeric expression you are monitoring for rate of change.

Update can be a logical expression or numeric constant. If you specify **Update** as a numeric constant, it creates an internal pulse timer with a pulse period of the specified time and a pulse duration of zero. See *Numeric Data Members* in Chapter 2, *Introduction*, of the *Getting Started with LookoutDirect* manual for information on entering time constants. If you specify **Update** as a logical variable, the variable should pulse at the frequency you want to use.

The **Update** expression triggers the calculation of a new rate-of-change based on the **Input** value at the prior **Update**, and the current **Input** value. The current **Input** value is then stored as the prior **Input** value for the next calculation. The **Update** period should be greater than the refresh rate of the incoming signal; or if the Input is generated directly by external I/O, the update data member generated by the PLC object should be used. If the **Update** period is less than the **Input** refresh rate, the rate of change calculation fluctuates erratically between zero and a high value.

Time unit is a numeric expression used as the basis for unit time on the Input signal. For instance, if the rate of change should be in feet per minute, the Input signal would be feet, and **Time unit** would be one minute (entered as 1:00). Typically the **Time unit** is one second (0:01), one minute (1:00), one hour (1:00:00), or one day (1:00:00:00). However, you can specify any unit, such as 5:23 (a rate of change in Input units per five minutes and 23 seconds).

Derivative Data Members

Table 2-13. Derivative Data Members

Data Member	Type	Read	Write	Description
(implicit)	numeric	yes	no	Rate of change

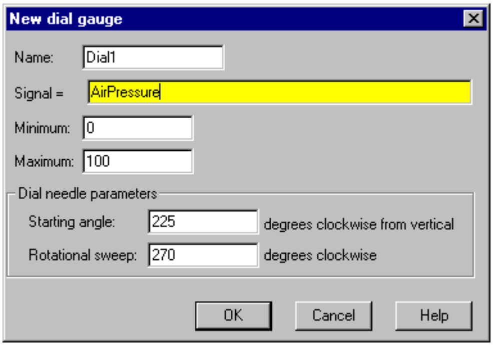
Comments Derivative performs the inverse function of Integral—you can theoretically run a signal through an Integral object and then a Derivative object (or vice versa) and you would end up with the original signal. (Discretization of the time calculations by the computer may cause the final and original signals to differ somewhat).

It is important to consider the resolution of the process variable measured by the PLC when determining the Update period for this object. For instance, if a pressure transmitter connected to a PLC only has a resolution of 0.5 psi and you want to measure rates down to 1 psi/minute, the Update pulse must be greater than 30 seconds even if the PLC is polled once per second (i.e., 0.5 psi/1 psi/min. = 30 sec.). For this application, the Update pulse should probably be about two minutes.

Related Objects *Integral*

DialGauge

The DialGauge object class displays a numeric signal as a sweeping needle on an analog gauge or dial.



New dial gauge

Name: Dial1

Signal = AirPressure

Minimum: 0

Maximum: 100

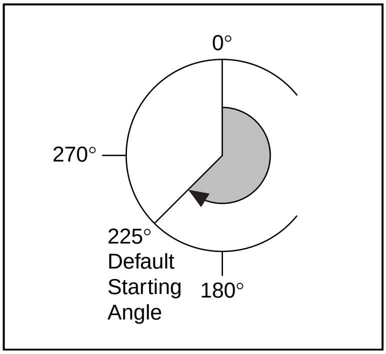
Dial needle parameters

Starting angle: 225 degrees clockwise from vertical

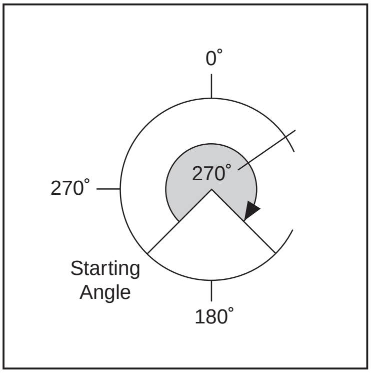
Rotational sweep: 270 degrees clockwise

Signal is a numeric expression.

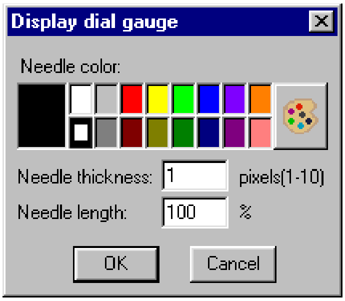
Starting angle indicates the position of the needle when the **Signal** is at its **Minimum** value. As shown here, you specify the starting needle position by counting the degrees clockwise from vertical.



Rotational sweep specifies the number of degrees clockwise that the needle will rotate as the **Signal** approaches the **Maximum** value. As shown in the diagram here, you count the **Rotational sweep** in degrees clockwise from the **Starting angle**.



After you specify the DialGauge definition parameters, Lookout*Direct* presents a display parameters dialog box, as shown here. You use this dialog box to specify needle color, thickness, and length.


Display dial gauge

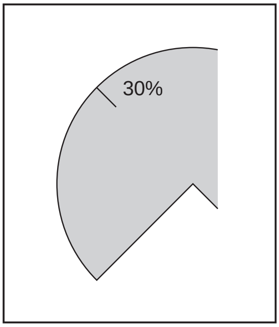
Needle color:

Needle thickness: 1 pixels(1-10)

Needle length: 100 %

Needle thickness defines how wide your needle will be. Thickness can range from one pixel (hairline) to 10 pixels wide.

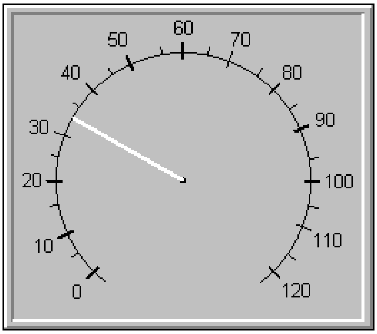
Needle length specifies the length of the needle as a percent of the radius. At 30 percent, for example, only the outer tip of the needle is visible—you cannot see the part of the needle closest to the origin. At 100 percent, the needle extends the full radius of the circle.



DialGauge Data Members

Table 2-14. DialGauge Data Members

Data Members	Type	Read	Write	Description
(implicit)	numeric	yes	no	Current value of signal parameter



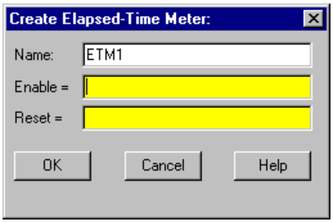
Comments The DialGauge object class only displays a needle. You may wish to enhance it with a corresponding scale or dial face as shown. You can create a scale or dial face by importing one from a graphics package. See *Creating Custom Graphics* in Chapter 2**,** *Graphics*, in the *LookoutDirect Developer's Manual* for more information.

Note If you choose to import a scale or dial face from an external package, you should use a bitmap instead of a metafile.This makes the display refresh cleaner when the needle changes position.

Related Objects *(expression)*, *Gauge*

ElapsedTime

ElapsedTime is an elapsed time meter or “hour meter” that totals the amount of time the **Enable** expression is on. If **Enable** is the logical constant `ON`, the meter reflects the time since the process was started. If a **Reset** expression is specified, the meter resets to zero the moment **Reset** transitions from off to on. The display always shows the elapsed time, and is updated approximately once per second.


Create Elapsed-Time Meter:

Name: ETM1

Enable =

Reset =

ElapsedTime Data Members

Table 2-15. ElapsedTime Data Members

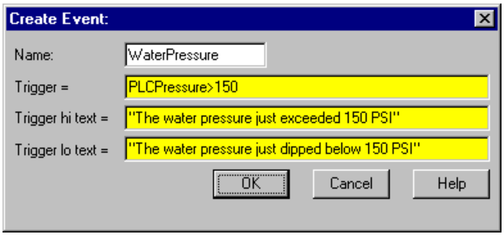
Data Member	Type	Read	Write	Description
(implicit)	numeric	yes	no	Total elapsed time—updated once per second, while meter is running

Comments ElapsedTime meters are used primarily to record the amount of time that individual pieces of equipment have been running. It is also straightforward to set up an alarm that sounds when a particular device has been operating for a certain time and needs routine servicing. The plant operator could then perform the service and reset the ElapsedTime meter with a pushbutton.

ElapsedTime can also record the amount of time a particular device is operated each day and you can record the resulting time to a daily Spreadsheet which you can then use to automatically reset the meter after the data is permanently recorded.

Event

Event is a flexible and powerful object class you can use to define event messages that are triggered based on a user-defined logical expression. Lookout*Direct* logs such events to the Citadel database and you can subsequently print and archive them. See Chapter 7, *Logging Data and Events*, in the *LookoutDirect Developer's Manual* for more information on logging events.



Create Event:

Name: WaterPressure

Trigger = PLCPressure>150

Trigger hi text = "The water pressure just exceeded 150 PSI"

Trigger lo text = "The water pressure just dipped below 150 PSI"

When the result of the **Trigger** logical expression transitions from FALSE to TRUE, it logs the result of the **Trigger hi text** expression as an event in the `EVENT.DAT` file. When the results of the **Trigger** expression transitions from TRUE to FALSE, it logs the results of the **Trigger lo text** expression as an event.

Event Data Members

Table 2-16. Event Data Members

Data Member	Type	Read	Write	Description
none	—	—	—	Event objects do not have data members

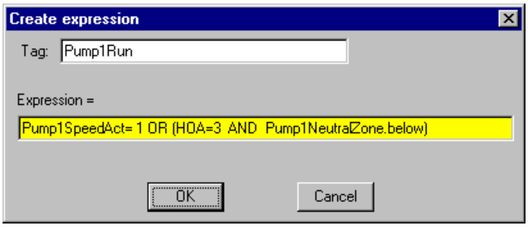
Comments For each event logged, Lookout*Direct* records the date and time, the name of the user currently logged on, and the expression text.

Although event messages are shown for both **Trigger hi text** and **Trigger lo text**, you do not have to include text in both fields.

(expression)

Named expressions, shown as (expression)s, are flexible, powerful real-time calculators. They create and calculate the result of spreadsheet-style formulas that include a mixture of constants and signals from other objects. There are over fifty built-in functions that you use in expressions, including logical, mathematical, statistical, text and trigonometric functions. See Chapter 2, *Expressions*, in the *LookoutDirect Developer's Manual* for more information on expressions and expression functions.

Named expressions can be short and simple, or extremely complex with several signal inputs, function calls, and multiple levels of parentheses. A single expression may incorporate text, logical, and numeric calculations. The variable type returned by the outermost function or operator in the expression determines the signal type generated by the expression.



Tag: Pump1Run

Expression = Pump1SpeedAct= 1 OR (HOA=3 AND Pump1NeutralZone.below)

Note You typically use (expression) objects when you need to define a unique condition that is used multiple times throughout your application.

When you define an (expression) object (as opposed to inserting an intrinsic expression), you create a unique name for your expression and can therefore reference the output signal generated from the expression in other expressions or objects. Instead of defining the same expression in many places, you create it one time and use its name any time you need this expression.

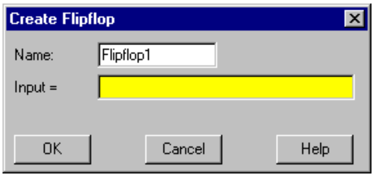
Note The expression may not express a **condition**.

Table 2-17. Expression Data Members

Data Member	Type	Read	Write	Description
(implicit)	numeric, logical, or text	yes	no	Value of expression. The variable type returned by the outermost function or operator in the expression determines the signal type generated by the expression.

Flipflop

Flipflop changes its logical output signal from on to off, or from off to on when the **Input** signal goes high. The output signal does not change when the signal goes low. **Input** is a logical expression.



Name: Flipflop1

Flipflop Data Member

Table 2-18. Flipflop Data Member

Data Member	Type	Read	Write	Description
(implicit)	logical	yes	no	Current state

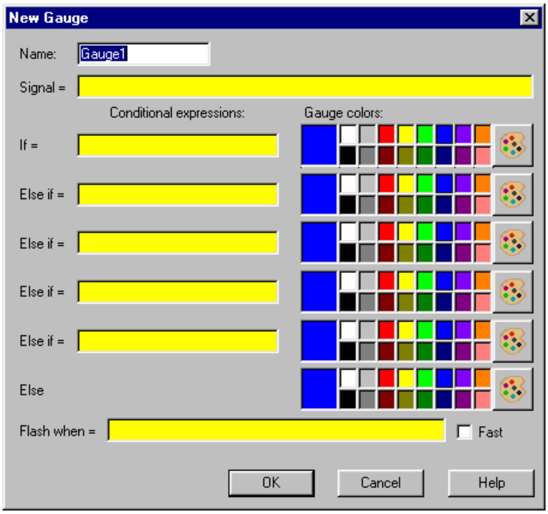
Comments Flipflop can be used to alternate the operation of two pumps, or when connected to a pushbutton, provides a pushbutton on/off control device.

Related Objects *LatchGate*

Gauge

Gauge displays the **Signal** expression in digital or bar graph format. Gauge display parameters change depending on the values of the **Conditional expressions**. Gauge determines which colors to display based on the order and current status of your conditional expressions. For instance, if several conditional expressions are true at once, Gauge displays the color associated with the first true expression.

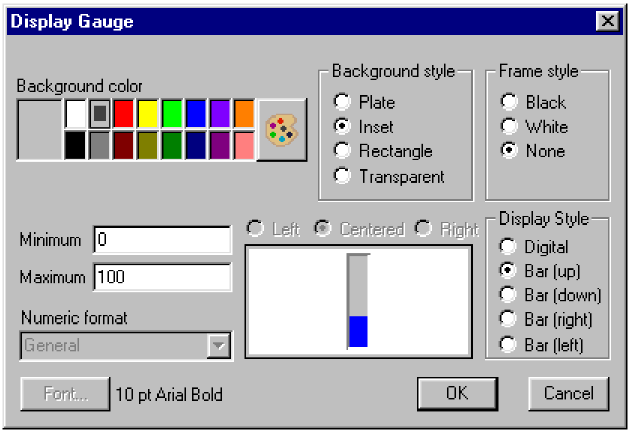
You can use the **Transparent** background style with numeric expressions and gauges displayed as bar graphs. This means you can have bar graphs with transparent backgrounds.



New Gauge

Name: Gauge1

Conditional expressions and **Flash when** are logical expressions while **Signal** is a numeric expression. The **Fast** option instructs the Gauge to flash faster when enabled than when disabled.



Display Gauge

Background color

Background style
○ Plate
● Inset
○ Rectangle
○ Transparent

Frame style
○ Black
○ White
● None

Minimum 0

Maximum 100

Numeric format General

○ Left ● Centered ○ Right

Display Style
○ Digital
● Bar (up)
○ Bar (down)
○ Bar (right)
○ Bar (left)

10 pt Arial Bold

Gauge Data Members

Table 2-19. Gauge Data Members

Data Member	Type	Read	Write	Description
(implicit)	numeric	yes	no	Numeric value of Gauge

Comments You should use a Gauge object when you need a bar graph or digital display to change colors and/or flash upon certain conditions. If you do not need either of these capabilities, you should display a bar graph or digital value with the **Insert»Expression…** command.

HyperTrend

A HyperTrend object displays a trend graph on a control panel. It plots any number of logical and numeric trend lines.

HyperTrends provide instant access to both real-time and historical data in a single graph. For each plot line, they combine both real-time and historical data into a seamless, contiguous trace of data. See *Citadel Historical Database* in Chapter 7, *Logging Data and Events*, in the *LookoutDirect Developer's Manual*.

Note There has been an important change in the way Lookout*Direct* logs data to the Citadel database to be displayed on the HyperTrend object.

In Lookout*Direct* 3.*xx*, creating a trend line in a HyperTrend object automatically logged the requested data to the database.

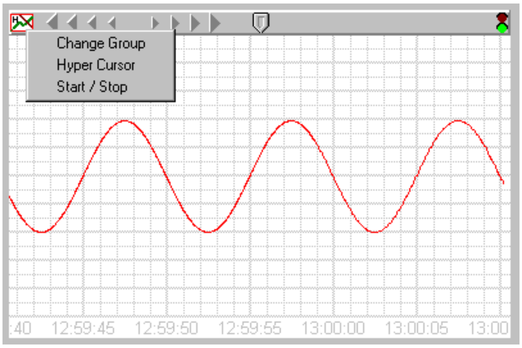
In Lookout*Direct* 4, you must select the Log to historical database option in the Edit Database dialog box for any data you want to display in a HyperTrend object before you can display the data. This is because logging must be configured in a Lookout*Direct* Server process, while trending is typically done in a separate Lookout*Direct* client process.

You can log information by connecting to a symbolic link, but only if the data source the symbolic link represents has had the Log to historical database option selected.

You can use HyperTrends to pan and zoom both the X axis and the Y axis, enabling dynamic adjustment of the vertical and horizontal resolutions of each plot line on the graph. Using this feature, you can, for example, zoom into a particular area of focus on the trend.

The graph scrolls from right to left, plotting current, real-time signals at the right end of the graph.

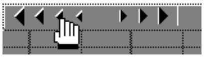
The icon in the upper left of the display accesses a menu you can use to change the group of traces being plotted, activate the cursor, and start or stop scrolling. The arrow-shaped buttons make it easy for you to scroll the trend graph forward and back in time. It provides instant access to data that has scrolled off the left end of the graph (that is, historical data stored in the Citadel database).



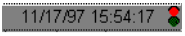
The button bar includes scroll arrows, a cursor button, date, time, and a stop and go light. Use the scroll arrows to move back and forth through time-the bigger the arrow button you select, the further the trend jumps in time. The scroll arrows also function much like a horizontal slider. Click on them and slide the mouse left and right while holding down the mouse button. The further you slide the cursor from dead center, the faster the trend scrolls in that direction.

Use the date and time indicators to choose a specific month, day, year, hour, minute, or second. If you click on the lower part of the hour, for example, it jumps back in time by one hour.

If you click on the upper part of the hour, it jumps ahead by one hour.



It works the same way for month, day, year, minute and second.



The stop & go light on the button bar is either red or green. If the light is green, it indicates the far right edge of the trend window displays the current time.

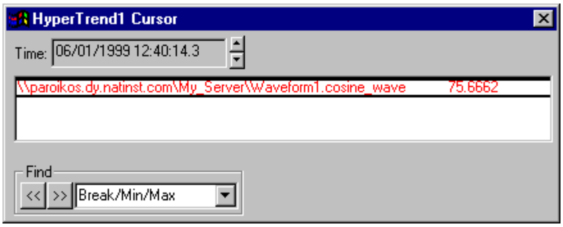
When you scroll back in time or if you click on the light when it is green, it changes to red, indicating that the trend is temporarily frozen in time. The date and time appears in the button bar indicating the exact time at the far right edge of the trend window. As you scroll back and forth through time, the data and time changes accordingly.

If you click on the light when it is red, the trend jumps back to current time and starts scrolling while plotting real-time values.

Note The Citadel database continues to log data no matter what state the HyperTrend is in. You do not lose any data when it is in "historical" mode (that is, when the HyperTrend is not scrolling in real-time).

When you click on the cursor button, a vertical cursor bar appears in the center of the graph along with an associated cursor dialog box. The dialog box indicates the value of each trend line at the current location of the cursor. As you drag the cursor bar left and right on the trend graph, the values in the pop-up change to reflect the new cursor location.

You can select how the trend line values are shown by choosing a format through the cursor control menu.


HyperTrend1 Cursor

Time: 06/01/1999 12:40:14.3

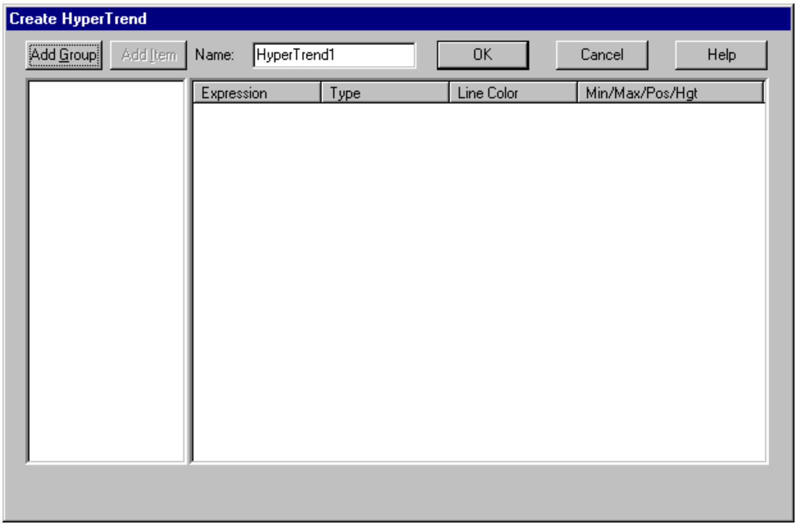
\\paroikos.dy.natinst.com\My_Server\Waveform1.cosine_wave 75.6662

Find: Break/Min/Max

Time indicates the current location of the cursor bar. The increment and decrement buttons beside the field move the cursor left and right in the trend graph. Choose the size of the incremental move by clicking on the desired portion of the date/time. The hour portion is selected in the previous example, so each time you click on the increment or decrement button, the cursor bar jumps ahead or back by an hour. It works the same way for any portion of the date and time.

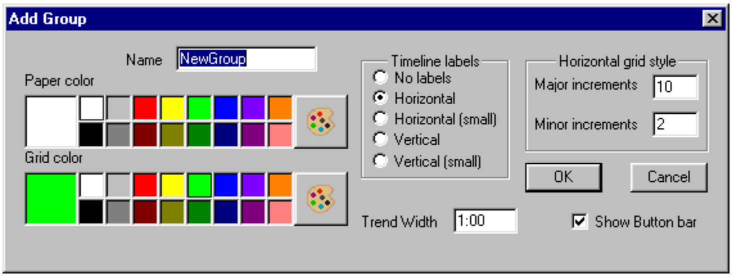
Use the **Find** combo box to search for a break in the trend line, a signal peak or valley, or a specific value. For example, you can find the last instance in which a process control limit value was exceeded. To find the last time a trend line crossed a specific value, choose the desired trend line by clicking on it in the list box, select **Value** in the **Find** combo box, enter the desired value, and clock on the scroll back button.

To create a hypertrend object, choose **Object»Create** or right-click on a process in the object explorer and select **New Object**. Select **HyperTrend**. The following dialog box appears.



Name: HyperTrend1

First you must create a group of trends. Double-click on the **AddGroup** button. The following dialog box appears.



Name NewGroup

Timeline labels:
○ No labels
● Horizontal
○ Horizontal (small)
○ Vertical
○ Vertical (small)

Major increments 10

Minor increments 2

Trend Width 1:00

☑ Show Button bar

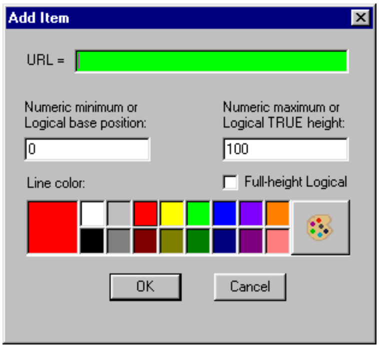
Configuring the group sets the appearance of the HyperTrend object when it displays traces from that group. You can set different paper and grid colors for different groups to help operators distinguish between them.

Give your group a name that clearly identifies it. Be sure to set the **Trend Width** for the period of time you want under observation for that group. Graphs may have a default width, or time span, of anywhere from two seconds to four years. The default **Trend Width** in the example dialog box indicates a time span of 1:00:00 or one hour. See *Numeric Data Members* in Chapter 2, *How LookoutDirect Works*, of the *Getting Started with LookoutDirect* manual for more information on entering time constants. After creating the HyperTrend object, you can make the trend width adjustable by connecting a numeric signal to the TrendWidth data member.

Major increments specifies the number of heavy horizontal grid lines on a trend graph. This value is independent of the range of any trend expressions.

Minor increments specifies the number of light horizontal grid lines between the major increment grid lines on a trend graph. This value is independent of the range of any trend expressions.

After you create a group, you must add individual items. Right-click on the group you want to add a trace to. Select **Add Item** (or click on the **Add Item** button). The following dialog box appears.



Add Item

URL =

Numeric minimum or Logical base position: 0

Numeric maximum or Logical TRUE height: 100

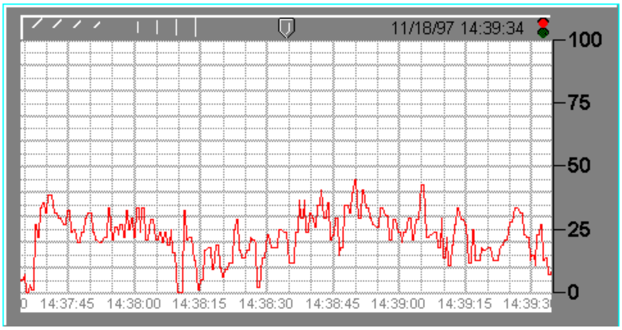
Line color:

□ Full-height Logical

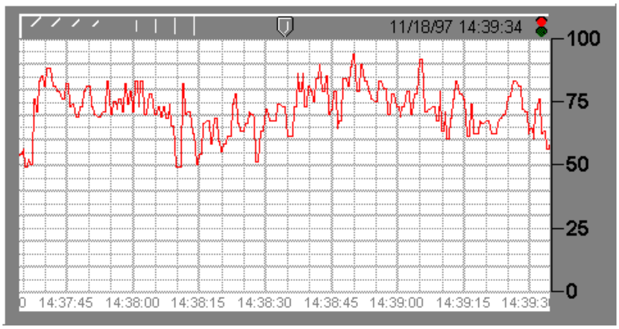
Right-clicking in the URL field displays the object selection dialog box covering all the registered computers running Lookout*Direct* on your network. Select the computer and process generating the value you want to display on your HyperTrend chart. See the URLs section for more information on this type of Lookout*Direct* connection.

Minimum and **Maximum** set the scale for the trace on your display. If you are charting a logical data member, position and height set the base display line and the height of the trace from that point when your signal goes TRUE. **Minimum** is the bottom of the graph while **Maximum** is the top of the graph, regardless of the range of the expression. These settings create an imaginary vertical scale and affect each expression independently.

For example, take two numeric expressions, both of which range from 0 to 50. Set the **Minimum** and **Maximum** to 0 and 100 on the first expression, and -50 and 50 on the second. The first expression plots in the bottom half of the chart while the second expression plots in the top half of the chart, even though they both fluctuate between 0 and 50.

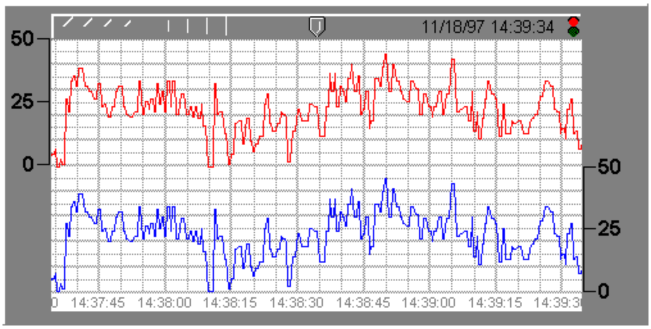


This figure shows the imaginary scale of the first expression (where min. = 0 and max. = 100). Because the expression ranges from 0 to 50, it is plotted in the bottom half of the graph.

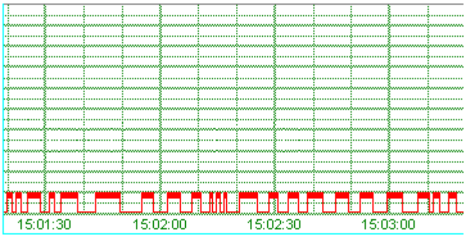


This figure shows the imaginary scale of the second expression (where min. = -50 and max. = 50). Because the expression ranges from 0 to 50, it is plotted in the top half of the graph.

When both expressions are entered on a single trend graph, you get the following effect. Notice the custom scales at either end of the graph.



If you select Logical for the expression type, the minimum and maximum settings changes to **Position** and **Height**. These two values now represent a number between 0% and 100%, and determine the baseline location of the trend line and its unit height when the expression goes TRUE.



Use **Insert»Scale** to label the values being charted according to your minimum and maximum settings.

Comments HyperTrend objects access data from the Citadel database. Think of them as windows into your historical database. See Chapter 5, *Developer Tour* in the *Getting Started with LookoutDirect* manual, and Chapter 7, *Logging Data and Events*, in the *LookoutDirect Developer's Manual*, for more information on specifying a point to be logged to the Citadel database.

HyperTrends are updated as quickly as once per second, depending on screen resolution, the size of the graph, and the trend width setting. Computers with slow display adapters may slow down considerably when you display a large trend graph. On slower computers with slow display cards (no graphics coprocessor), consider limiting the size of your HyperTrends to less than one fourth the screen size.

HyperTrend Data Members

Table 2-20. HyperTrend Data Members

Data Members	Type	Read	Write	Description
ActiveGroup	numeric	yes	yes	Sets which of the HyperTrend trace groups is active.
Enable1– Enable 999	logical	yes	yes	When TRUE, the identified trend line is visible. When FALSE, the trend line hidden. The default value is TRUE. Trend lines are identified by the group number followed by the item number, so Enable1.2 would refer to item 2 in group 1.
Height1 – Height999	numeric	yes	yes	Specifies the amplitude or height of the identified trend line (distance from baseline) when the logical expression goes TRUE. Height should be between 2 and (100 minus position). Trend lines are identified by the group number followed by the item number, so Height1.2 would refer to item 2 in group 1.
Max1 – Max999	numeric	yes	yes	Specifies the top of the graph for the identified numeric trend line (the value of the trended line when it is at 100 percent of the Y axis). Trend lines are identified by the group number followed by the item number, so Max1.2 would refer to item 2 in group 1.
Min1 – Min999	numeric	yes	yes	Specifies the bottom of the graph for the identified numeric trend line (the value of the trended line when it is at zero percent of the Y axis). Trend lines are identified by the group number followed by the item number, so Min1.2 would refer to item 2 in group 1.

Table 2-20. HyperTrend Data Members (Continued)

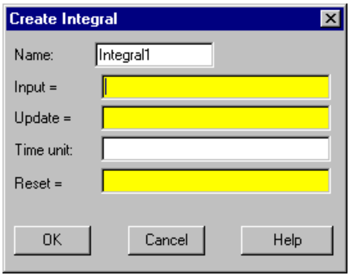
Data Members	Type	Read	Write	Description
Pos1 – Pos999	numeric	yes	yes	Specifies the baseline location of the identified logical trend line. Baseline position should range should be 1–98. (pos1 is associated with trend line 1) Trend lines are identified by the group number followed by the item number, so Pos1.2 would refer to item 2 in group 1.
TrendWidth	numeric	yes	yes	Specifies the span of time that the X axis covers.
UseButtonBar	logical	yes	yes	When TRUE, the HyperTrend button bar becomes visible on the control panel. When FALSE, it is invisible. The default value is TRUE.
Visible	logical	yes	yes	When TRUE, the HyperTrend becomes visible on the control panel. When FALSE, it is invisible. The default value is TRUE.

Converting Lookout*Direct* 3.*xx* HyperTrends to Lookout*Direct* 4

If you convert a process created in a version of Lookout*Direct* earlier than 4.0 which contains a HyperTrend object that is displaying a complex expression, Lookout*Direct* automatically creates a named (expression) for the converted HyperTrend to chart. To maintain your legacy HyperTrend objects you must use or replace these (expression) objects.

Integral

Integral is a totalizer—it totals the numeric **Input** signal. This class is typically used to total a measured flow rate.



Input is the numeric expression that you want to totalize or integrate.

Update can be a logical expression or numeric constant. If you specify **Update** as a numeric constant, it creates an internal Pulse timer with a pulse period of the specified time and a pulse duration of zero. See *Numeric Data Members* in Chapter 2, *How LookoutDirect Works*, of the *Getting Started with LookoutDirect* manual for information on entering time constants. If you specify **Update** as a logical variable, the variable should pulse at the desired frequency.

The **Update** expression extrapolates an interim total based on the current total and the most recent Input value. The interim total is then sent out as the output. The total is calculated using the trapezoidal numeric integration technique, and the total is corrected any time the incoming signal is refreshed.

Time unit is a numeric expression used as the basis for unit time on the Input signal. For instance, if the Input rate is in units of gallons per minute, the Time unit should be entered as one minute (1:00) so the totalized flow is in gallons. Typically the Time unit is one second (0:01), one minute (1:00), one hour (1:00:00), or one day (1:00:00:00). However, you can specify any unit, such as 5:23 (a rate of change in Input units per five minutes and 23 seconds).

Reset is a logical expression that resets the totalizer value to zero upon transition from OFF to ON.

Note Integral does not have a display parameters dialog box. You can easily display the result of the Integral output signal by referencing its data member in an expression.

Integral Data Members

Table 2-21. Integral Data Members

Data Member	Type	Read	Write	Description
(implicit)	numeric	yes	no	Totalized value

Comments The **Update** pulse forces the calculated total to continue changing between Input signal updates. For example, if a remote RTU that is monitoring a flow rate is polled every ten minutes, the **Update** pulse could be set at five seconds so the operator can watch the totalized flow continue to change on the screen as an extrapolated value. The corrected totalized value is calculated any time the **Input** signal refreshes—in this case, every ten minutes.

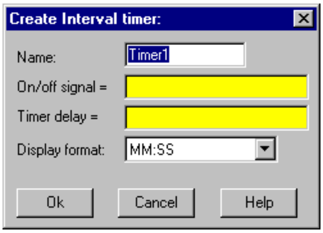
If totalized values are logged to a spreadsheet on a daily basis, for example, and the total should be reset at the end of every day, use the update pulse generated by the Spreadsheet object to reset the total—this guarantees that the total is recorded before the totalizer is reset. The example on the previous page totalizes the hourly flow for permanent data logging by a spreadsheet object named HourlySheet. Notice that the spreadsheet update pulse `HourlySheet.logged` is used to reset the totalizer.

Related Objects *Accumulator*, *Counter*, *Derivative*

Interval

Interval is an adjustable delay timer. When **On/off signal** transitions to on, its output turns on and the **Timer delay** begins to count down. At the end of the delay countdown, the output signal turns OFF. If **On/off signal** is dropped at any time, the output signal turns OFF, and the timer is reset.

Timer delay can range from 0.0 seconds to several years, and the effective resolution is 0.1 seconds over the entire range. The timer display digitally shows the time delay remaining. It is updated approximately once per second. If the **On/off signal** is low, or the time delay period has expired, the timer display shows `OFF`.



Create Interval timer:

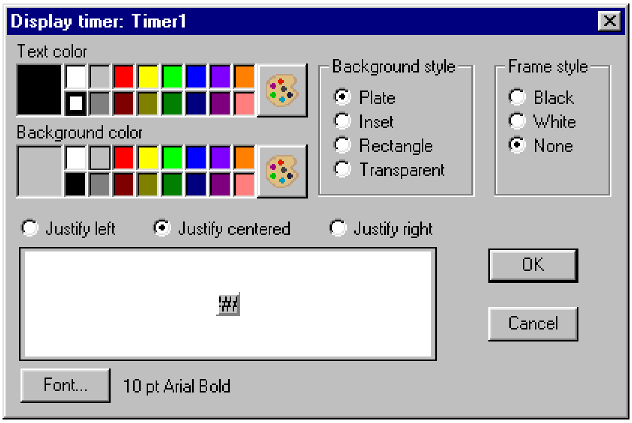
Name: Timer1

On/off signal =

Timer delay =

Display format: MM:SS

The **On/off signal** is a logical expression while **Timer delay** is a numeric expression. Normally, this is a simple time constant such as 0:20 (twenty seconds). See *Numeric Data Members* in Chapter 2, *How LookoutDirect Works*, of the *Getting Started with LookoutDirect* manual for information on entering time constants.



Display timer: Timer1

Background style: ● Plate ○ Inset ○ Rectangle ○ Transparent

Frame style: ○ Black ○ White ● None

○ Justify left ● Justify centered ○ Justify right

Font... 10 pt Arial Bold

Interval Data Members

Table 2-22. Interval Data Members

Data Member	Type	Read	Write	Description
(implicit)	logical	yes	no	Logical timer value

Comments The Interval timer can be used to enforce a maximum run time for a pump.

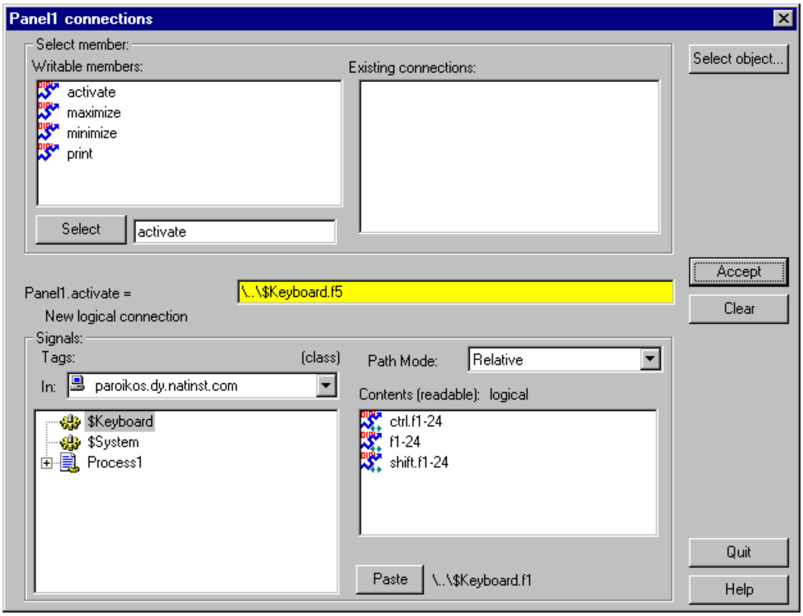
Related Objects *DelayOff*, *DelayOn*, *OneShot*, *Pot*, *TextEntry*

$Keyboard

$Keyboard is a global object. Its data members represent the keyboard function keys. Unlike other object classes in which you can create several objects of the same class, you cannot create or delete $Keyboard objects, but you can use the one supplied.

You can use the $Keyboard global object to perform such functions as calling a particular control panel, activating a batch sequence, or acknowledging alarms by pressing a key.

Think of $Keyboard data members (which represent function keys on the keyboard) as Lookout*Direct* pushbuttons. Just as you can connect a pushbutton to the activate data member of a panel, you can also connect a $Keyboard data member to the activate data member of a panel. Such a connection is shown in the following illustration.



Writable members: activate, maximize, minimize, print

Select: activate

Panel1.activate = \..\$Keyboard.f5

Path Mode: Relative

In: paroikos.dy.natinst.com

Tags: $Keyboard, $System, Process1

Contents (readable): logical

ctrl.f1-24
f1-24
shift.f1-24

Paste \..\$Keyboard.f1

The logical expression, `$Keyboard.F1` calls up Panel1 any time a user presses the F1 key on the keyboard. Similar connections could be made to other panels. You can easily connect a different panel to each function key.

Just as easily, you can connect a function key to a batch process trigger. When the key is pressed, (that is, when the `$Keyboard` data member goes TRUE) the batch is activated—reading batch ingredients from a recipe object, opening and closing valves, starting mixers, bottling finished material, and so on.

You might also connect a function key to `$Alarm.ack`. This would enable users to acknowledge alarms through a single keystroke.

$Keyboard Data Members

$Keyboard has 72 readable data members. Each data member represents a unique key sequence, described in the following table.

Table 2-23. $Keyboard Data Members

Data Member	Type	Read	Write	Description
Ctrl-F1 - Ctrl-F24	logical	yes	no	Each of these 24 data members represent a function key, F1 – F24—when pressed in conjunction with the Ctrl key. A given data member returns logical TRUE when the Ctrl key and function key are pressed together and FALSE when the keys are released.
F1 - F24	logical	yes	no	Each of these 24 data members represent a function key, F1 – F24. A given data member returns a logical TRUE when its associated function key is pressed and FALSE when the key is released.
Shift-F1 - Shift-F24	logical	yes	no	Each of these 24 data members represent a function key, F1 – F24—when pressed in conjunction with the Shift key. A given data member returns logical TRUE when the Shift key and function key are pressed together and FALSE when the keys are released.

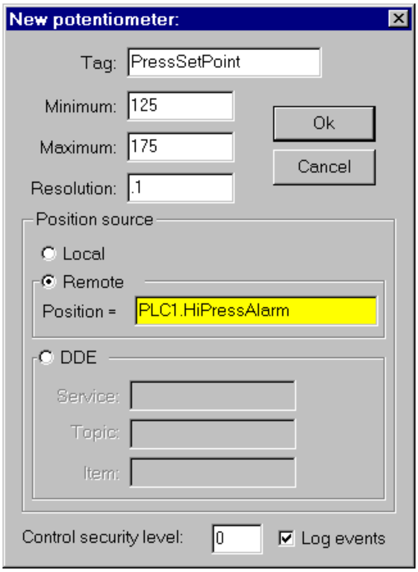
Comments $Keyboard function keys are global in nature. Any time F1 is pressed, the `$keyboard.F1` signal goes TRUE—regardless of what panel the user is looking at. If you want a function key to be unique from one control panel to the next, use the Panel object class function key data member. See *Panel* object class definition for more information.

Related Objects *L3Pushbutton, Pushbutton*

L3Pot

L3Pot is a potentiometer that you use to change numeric setpoint values. You can display pots on a control panel as a knob, vertical slider, horizontal slider, increment/decrement pushbuttons, or digital entry. You can also use pots as multiple-position switches.

If you change the background color of a panel and add a Pot object displayed as a slider, its color is always gray. To change the background color of a Pot to match your panel, select the Pot object, then pick **Change»Background Color** from the menu.



New potentiometer:

Tag: PressSetPoint

Minimum: 125

Maximum: 175

Resolution: .1

Position source

○ Local

● Remote

Position = PLC1.HiPressAlarm

○ DDE

Service:

Topic:

Item:

Control security level: 0

☑ Log events

Minimum is the lowest value signal the Pot will generate.

Maximum is the highest value signal the Pot will generate.

Resolution is the smallest increment of change, or detent spacing the Pot supports.

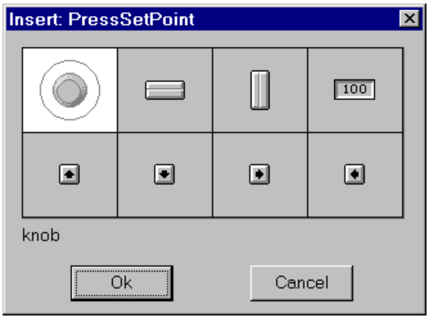
Position source determines where the value of the Pot resides. **Local** indicates the value of the Pot lies within the object itself—on the control panel.

Remote pots get their values from a remote source, often the register on a controller they are connected to. Adjusting the Pot changes the value in the register, and changing the value in the register adjusts the Pot. In effect, the Pot is tracking a remote value. This is especially useful when you want to prevent Lookout*Direct* from changing the value of setpoints or registers upon initial startup, or reconnection of lost communication. When you use this style Pot you are creating a kind of looped signal. Half the loop is formed when you connect the controller register to the Pot with the **Position** expression, while the second half is formed when you connect the Pot output signal to the controller register. **Position** is a numeric expression. Do not forget to complete the second half of the loop with the **Object»Edit Connections…** command.

Much like Remote Pots, **DDE** (Dynamic Data Exchange) Pots get their values from a remote source. This could be a cell in a spreadsheet, another DDE aware application, or a second copy of Lookout*Direct* running on the network. The last DDE parameters used on any object automatically become the default values for any new DDE object. See Chapter 5, *Dynamic Data Exchange*, of the *LookoutDirect Developer's Manual* for more information on Service, Topic, and Item parameters.

Control security level specifies the minimum security level operators must have to gain access to this individual object, and thus control it.

The **Log events** option creates a permanent audit trail for the object—who did what and when. All adjustments of the Pot are logged to disk, including the time of the adjustment, the operator account name, and what adjustment was made. See Chapter 7, *Logging Data and Events*, in the *LookoutDirect Developer's Manual* for more information on event logging.



Note You can modify the background color on vertical and horizontal sliders with the **Change»Background color…** menu command. You can modify the font and font color of digital pots using Change commands.

L3Pot Data Members

Table 2-24. L3Pot Data Members

Data Member	Type	Read	Write	Description
(implicit)	numeric	yes	no	Current value
decrement	logical	no	yes	When this data member value transitions from FALSE to TRUE, the implicit value of the Pot object decreases by the Pot **Resolution** amount.
enable	logical	no	yes	If TRUE (the default), enables DDE. If FALSE, disables DDE. The default value is on. This input is ignored for non-DDE TextEntry objects.
enterValue	logical	no	yes	Under specific circumstances, when this value transitions from FALSE to TRUE, pops up a **Enter new value** dialog box for an operator to use in entering a value for the pot. See the note following the table for more detailed information. The enterValue data member is designed for use under unusual circumstances, in particular when pointing devices are not available on a computer running Lookout*Direct*. Because of its unusual operation, this data member should not be used unless it is necessary for hardware reasons.
increment	logical	no	yes	When this data member value transitions from FALSE to TRUE, the implicit value of the Pot object increases by the **Resolution** amount.
visible	logical	no	yes	When FALSE, the Pot object cannot be seen on the display panel. When TRUE, the Pot can be seen and controlled.

Note When the **`enterValue`** input transitions from **`FALSE`** to **`TRUE`**, and
if the Pot is visible,
if Lookout*Direct* is not in edit mode, and
if the Pot has at least one digital display,

The **Enter new value** dialog box pops up so an operator can input a value, just as if he had clicked on the digital display.

The numeric format and position used for the dialog box are based on the digital display for a pot.

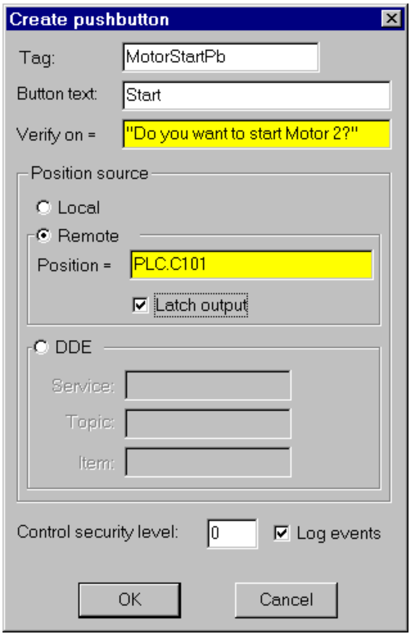
Even if the panel containing the Pot digital display is inactive, the **Enter new value** dialog box will pop up. You can prevent this by predicating the **`enterValue`** input on the panel's **`active`** data member.

Comments Potentiometers are one of the most common control objects used in process controls. Using Pots, a plant operator can make setpoint changes with the mouse. L3Pots also work well as HOA switches. To create an HOA switch with a Pot, specify the minimum as 1, the maximum as 3, and the resolution as 1.

The increment and decrement data members enable quick connection of pot objects to $Keyboard and Panel function keys, and screen Pushbuttons. These are often used to control Pot objects when Lookout*Direct* is running on an industrial PC platform that has restricted or no mouse functionality.

L3Pushbutton

L3Pushbutton generates a logical signal for receipt by other objects. A pushbutton changes state when you position the cursor over it and press the mouse button, trackball, touchscreen, or space bar. The pushbutton remains depressed and the output signal remains high until you release the button. If a **Verify on** message is defined, the operator must first acknowledge the message, then the output signal goes high, *but only momentarily*.



Create pushbutton

Tag: MotorStartPb

Button text: Start

Verify on = "Do you want to start Motor 2?"

Position source

○ Local

● Remote

Position = PLC.C101

☑ Latch output

○ DDE

Service:

Topic:

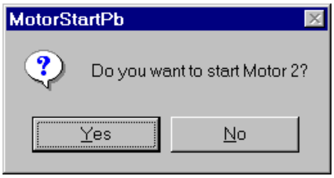
Item:

Control security level: 0

☑ Log events

Button text displays the specified text on the pushbutton.

Use **Verify on** to create a dynamic text expression to be displayed in a message dialog box. See Chapter 6, *Security*, in the *LookoutDirect Devloper's Manual* for more information on security.



MotorStartPb

Do you want to start Motor 2?

Position source determines where the value of the pushbutton resides. **Local** indicates the value of the pushbutton lies within the pushbutton itself—on the control panel. If the pushbutton is not depressed its signal is OFF, if depressed its signal is ON.

Remote pushbuttons get their values from a remote source, often the register in a controller they are connected to. Depressing the pushbutton changes the status of the register, and changing the status of the register depresses the pushbutton.

The **Remote** option is especially useful when you want to prevent Lookout*Direct* from changing the value of setpoints or registers upon initial startup, or reconnection of lost communication. When you use this style of pushbutton you are creating a sort of looped signal. Half the loop is formed when you connect the controller register to the pushbutton with the Position expression, while the second half is formed when you connect the pushbutton output signal to the controller register. Position is a logical expression. Do not forget to complete the second half of the loop with the **Object»Edit Connections…** command.

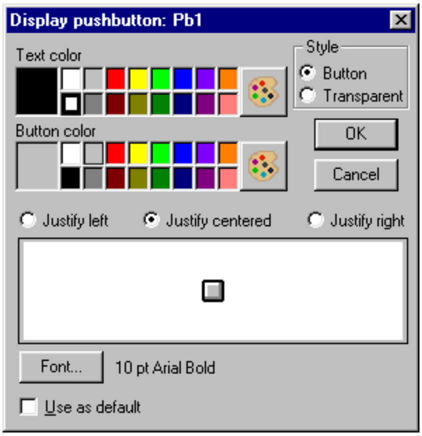
When you select the **Remote** option, you can choose whether or not the pushbutton latches its output. The **Latch output** check box configures Lookout*Direct* for controlling a latching-relay.

When a user clicks on a pushbutton that has latching selected, the pushbutton remains depressed, sending an ON signal (TRUE or high) until the Remote Position signal turns ON. Assume for example that an operator clicks on `MotorStartPb`, configured previously. The pushbutton remains pushed in, sending a TRUE signal, until PLC.C101 goes TRUE. As soon as PLC.C101 goes TRUE, the pushbutton releases.

Much like Remote pushbuttons, **DDE** (Dynamic Data Exchange) pushbuttons get their values from a remote source. This could be a cell in a spreadsheet, another DDE aware application, or a second copy of Lookout*Direct* running on the network. See Chapter 5, *Dynamic Data Exchange*, in the *LookoutDirect Developer's Manual* for information on **Service**, **Topic**, and **Item** parameters.

Control security level specifies the minimum security level operators must have to gain access to this individual object, and thus control it. See Chapter 6, *Security,* in the *LookoutDirect Developer's Manual* for more information on security.

The **Log events** option creates a permanent audit trail for the object—who did what and when. Any depression of the pushbutton is recorded to disk, including the time the button was depressed, and the operator’s account name. See Chapter 7, *Logging Data and Events*, in the *LookoutDirect Developer’s Manual* for more information on logging events.



Display pushbutton: Pb1

Style: ● Button ○ Transparent

○ Justify left ● Justify centered ○ Justify right

10 pt Arial Bold

□ Use as default

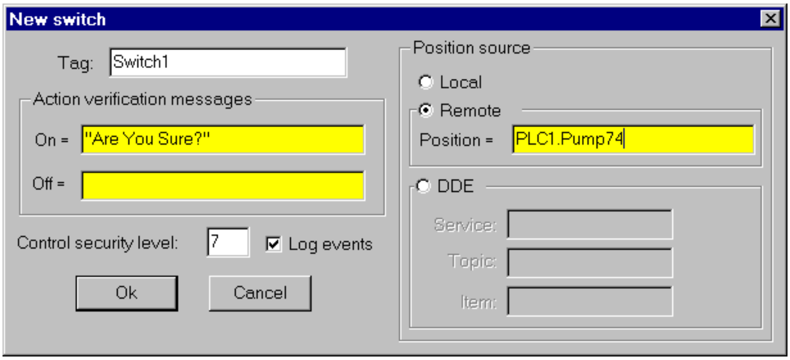
L3Pushbutton Data Members

Table 2-25. L3Pushbutton Data Members

Data Members	Type	Read	Write	Description
(implicit)	logical	yes	no	Value of object (TRUE when button is depressed)
enable	logical	no	yes	If TRUE (the default), enables DDE. If FALSE, disables DDE. The default value is on. This input is ignored for non-DDE TextEntry objects.
visible	logical	no	yes	When FALSE, the pushbutton cannot be seen on the display panel. When TRUE, the button can be seen and controlled.

L3Switch

L3Switch generates a logical signal for receipt by other objects. Switches change state when you click on them with a mouse button, trackball, touchscreen, or space bar on your keyboard.



Tag: Switch1

On = "Are You Sure?"

Off =

Control security level: 7

☑ Log events

○ Local

● Remote

Position = PLC1.Pump74

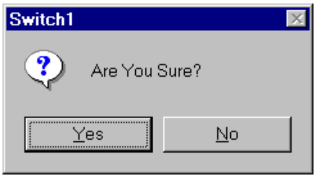
○ DDE

Service:

Topic:

Item:

Use **Action verification messages** to create dynamic text expressions to be displayed in message dialog boxes. See Chapter 6, *Security*, in the *LookoutDirect Developer's Manual* for more information on security.



Are You Sure?

Position source determines where the value of the switch resides. **Local** indicates the value of the switch lies within the object itself—on the control panel. If the switch is up the signal is ON, if down the signal is OFF.

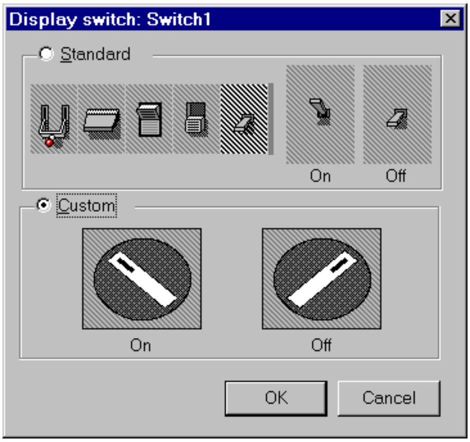
Remote switches get their values from a remote source, often the register on a controller they are connected to. Flipping the switch changes the status of the register, and changing the status of the register flips the switch.

The **Remote** option is especially useful when you want to prevent Lookout*Direct* from changing the value of setpoints or registers upon initial startup, or reconnection of lost communication. When you use this style switch, you are creating a sort of looped signal. Half the loop is formed when you connect the controller register to the switch with the **Position** expression, while the second half is formed when you connect the switch output signal to the controller register. Notice **Position** is a logical expression. Do not forget to complete the second half of the loop with the **Object»Edit Connections…** command.

Much like Remote switches, **DDE** (Dynamic Data Exchange) switches get their values from a remote source. This could be a cell in a spreadsheet, another DDE aware application, or a second copy of Lookout*Direct* running on the network. See Chapter 5, *Dynamic Data Exchange*, in the *LookoutDirect Developer's Manual* for more information on **Service**, **Topic**, and **Item** parameters.

Control security level specifies the minimum security level operators must have to gain access to this individual object, and thus control it.

The **Log events** option creates a permanent audit trail for the object—who did what and when. All adjustments of the switch are logged to disk, including the time the switch was flipped, the operator's account name, and the direction the switch was flipped. See Chapter 7, *Logging Data and Events*, in the *LookoutDirect Developer's Manual* for more information on event logging.


Display switch: Switch1

○ Standard

● Custom

You can replace the standard switch types with custom graphic symbols. If you decide to use custom graphics, you must specify both symbol parameters, **On** and **Off**. See Chapter 2, *Graphics*, in the *LookoutDirect Developer's Manual* for more information on creating custom graphic symbols and the use of Transparent pixels.

L3Switch Data Members

Table 2-26. L3Switch Data Members

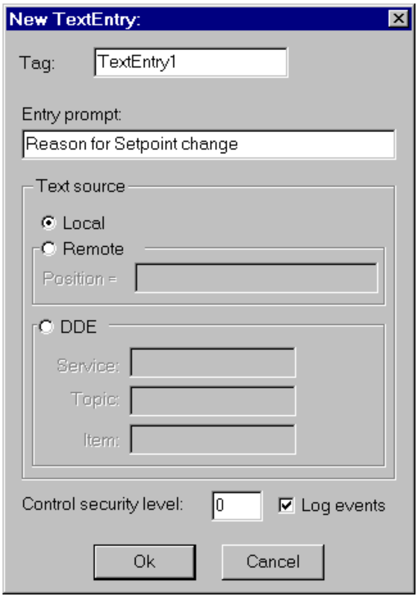
Data Members	Type	Read	Write	Description
(implicit)	logical	yes	no	L3Switch Position
enable	logical	no	yes	If TRUE (the default), enables DDE. If FALSE, disables DDE. The default value is ON. This input is ignored for non-DDE L3TextEntry objects.
visible	logical	no	yes	When FALSE, the switch object cannot be seen on the display panel. When TRUE, the switch can be seen and controlled.

Comments If a switch with more than two positions is needed, use a Radiobutton object instead.

Related Objects *L3Pushbutton, L3Pot, Pushbutton, Pot*

L3TextEntry

With L3TextEntry you can manually enter textual notes with the keyboard. These notes may contain any combination of numeric and alphanumeric characters; however, the result of your entry is converted to a text value. Just like any other text expression in Lookout*Direct*, your note can be logged to disk, connected to other data members that accept text signals, and so on. The note is saved and displayed as a single line entry—you cannot embed carriage returns into the message.



New TextEntry:

Tag: TextEntry1

Entry prompt: Reason for Setpoint change

Text source

● Local

○ Remote

Position =

○ DDE

Service:

Topic:

Item:

Control security level: 0

☑ Log events

Entry prompt is the text that appears at the top of the text entry dialog box when an operator selects the text entry pushbutton.

Text source determines where the user-entered text resides. **Local** indicates the user-entered text lies within the object itself—on the control panel.

Remote indicates that the user-entered text resides in a remote source, such as a text expression or another TextEntry object.

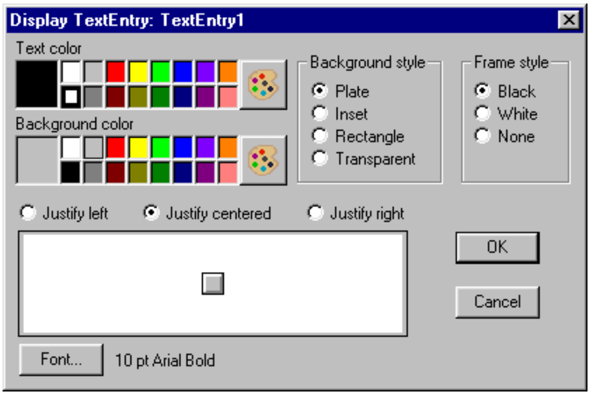
Much like **Remote** TextEntry objects, **DDE** TextEntry objects get their values from a remote source. This is the option you use to tie the text to a cell in a spreadsheet, a database lookup table, or any DDE aware application—including a second copy of Lookout*Direct* running on the network. See Chapter 5, *Dynamic Data Exchange*, in the *LookoutDirect Developer's Manual* for more detailed information on **Service**, **Topic** and **Item**.

Note The last DDE parameters used on any object automatically become the default values for any new DDE object.

Control security level specifies the minimum security level operators must have to gain access to this individual object, and thus control it.

The **Log events** option creates a permanent audit trail for the object—who did what and when. When selected, all text entries in this object are logged to disk. Each entry includes the time of the entry, the operator's account name, and what entry was made. See Chapter 7, *Logging Data and Events*, in the *LookoutDirect Developer's Manual* for more information on event logging.

Lookout*Direct* presents the following display parameters dialog box after you define the object. It lets you define the text font and presentation style.



Display TextEntry: TextEntry1

Background style: ● Plate ○ Inset ○ Rectangle ○ Transparent

Frame style: ● Black ○ White ○ None

○ Justify left ● Justify centered ○ Justify right

Font... 10 pt Arial Bold

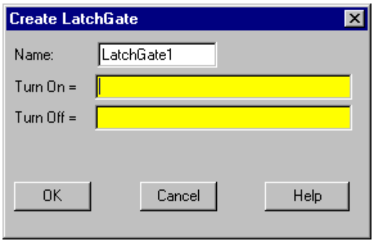
L3TextEntry Data Members

Table 2-27. L3TextEntry Data Members

Data Members	Type	Read	Write	Description
(implicit)	text	yes	no	Current, user-entered text
enable	logical	no	yes	If TRUE (the default), enables DDE. If FALSE, disables DDE. The default value is ON. This input is ignored for non-DDE TextEntry objects.

LatchGate

LatchGate is latched on and off by two incoming signals. It retains the state of the signal that most recently went high, regardless of the state of the other signal. When the **Turn Off** signal transitions from OFF to ON, the LatchGate output goes OFF until the **Turn On** signal transitions from OFF to ON. The output signal does not change when either incoming signal transitions from ON to OFF. Both **Turn Off** and **Turn On** are logical expressions.


Create LatchGate

Name: LatchGate1

Turn On =

Turn Off =

LatchGate Data Members

Table 2-28. LatchGate Data Members

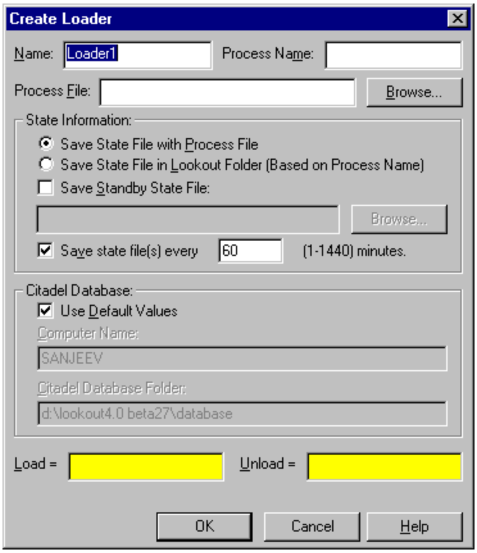
Data Member	Type	Read	Write	Description
(implicit)	logical	yes	no	Logical output signal value

Comments Two pushbuttons connected to the **Turn On** and **Turn Off** expressions of a LatchGate create pushbutton start/stop controls for a pump or other device.

Related Objects *Flipflop*

Loader

Use the Lookout*Direct* Loader to load or unload a Lookout*Direct* process in response to a logical trigger. Using a Loader object in conjunction with a Monitor object is how you create failover redundancy with Lookout*Direct*. See Chapter 10, *Redundancy*, in the *LookoutDirect Developer's Manual* for more information on redundancy.


Create Loader

Name: Loader1 Process Name:

Process File:

State Information:

● Save State File with Process File

○ Save State File in Lookout Folder (Based on Process Name)

□ Save Standby State File:

☑ Save state file(s) every 60 (1-1440) minutes.

Citadel Database:

☑ Use Default Values

Computer Name: SANJEEV

Citadel Database Folder: d:\lookout4.0 beta27\database

Load = Unload =

You may notice that the information in the **State Information** and **Citadel Database** sections of this dialog box are the same as those areas in the **Create Process** dialog box and, in the case of the **Citadel Database** section, the **System Options** dialog box. Just as the **Citadel Database** section of the **Create Process** dialog box overrides the system database location settings for a particular process, any settings you make in the Loader object for database location or saving of state files override both system defaults and whatever settings you may have made when you created the process being loaded.

Process Name is the name that the process runs under when you open the **Process File**. You can use a process name other than the one used when the process file loaded was created.

Process File is the name of the file this Loader object will load when activated. You must enter a complete path to the file. You can specify a file from another computer on your network, but you must map the location on the remote computer as a network drive on your own computer first.

The **State Information** section of this dialog box lets you set where Lookout*Direct* saves state files for the process just loaded, and under what name the files are saved.

Select **Save State File with Process File** to save the state file in the location where the process file was opened.

Select **Save State File in Lookout*Direct* Folder** to save the state file in the Lookout*Direct* folder of the copy of Lookout*Direct* your are currently running. The state file name is the same as the process name.

Select the **Save Standby State File** checkbox to save one or more extra copies of the state file in a location of your choosing. Enter a complete path, including state file name, to each location you want to save a state file. If you are saving the state file to more than one backup or alternative location, separate the paths with the vertical bar (`|`) operator symbol.

Check the **Save State File(s) every *`NNNN`* (1-1440) minutes** option to set the frequency with which Lookout*Direct* saves the state file. The Lookout*Direct* default is 60 minutes.

The **Citadel Database** section sets the location of the Citadel database that Lookout*Direct* logs data to for the process you load. If you check the **Use Default Values** checkbox, Lookout*Direct* uses the default location set in the **System Options** dialog box of any instance of Lookout*Direct* running the process.

If you enter a computer name and a path on that computer to a specific folder, Lookout*Direct* logs data to that location on that computer, no matter what computer is running the process.

To designate a specific computer and path for your process to log data to, enter the fully qualified network name for the target computer in the **Computer Name** field, and the complete path to the target database directory in the **Citadel Database Folder** field.

Load is the logical signal you use to activate the Loader and load the process.

Unload is the logical signal you use to activate the Loader and unload the process you loaded earlier.

The Loader can only load one process at a time, and can only load a process to run inside the instance of Lookout*Direct* currently running the process that contains the Loader.

To load multiple processes with one trigger, you must use multiple Loader objects.

Loader Data Members

Table 2-29. Loader Data Members

Data Member	Type	Read	Write	Description
SaveWithProcess	logical	yes	no	When TRUE, Lookout*Direct* saves a state file in the same location as the process file it contains the state for.
DatabaseComputer	text	yes	no	Sets the computer on which Lookout*Direct* saves Citadel database files.
DatabaseFolder	text	yes	no	Sets the folder in which Lookout*Direct* saves Citadel database files.
Failure	logical	yes	no	Monitor this data member to be alerted if a load attempt fails.
Failuretext	text	yes	no	Returns the reason for a failure to load the process designated by the loader.
Load	logical	yes	yes	Triggers the loader to load the designated process file.
ProcessFile	text	yes	no	Name of the process the loader loads when activated.
ProcessName	text	yes	no	Specifies the process name a loaded process runs under.
SavePeriod	numeric	yes	no	Sets how often, in minutes, that Lookout*Direct* saves the state file for the loaded process.
SaveWithLookout*Direct*	logical	yes	no	When TRUE, Lookout*Direct* saves the state file in the Lookout*Direct* directory, using `ProcessName` as the name for the state file as well.

Table 2-29. Loader Data Members

Data Member	Type	Read	Write	Description
StandbyFile	text	yes	no	Complete path to the location in which you want Lookout*Direct* to save copies of the state file.
Unload	logical	yes	yes	Triggers the loader to unload the designated process.

Loader Error Messages

Load: No process file specified

Specify a process file in the **ProcessFile** field.

Load: No process name specified

Name your process in the **ProcessName** field.

Load: Process already exists: *loadername.processname*

The process specified by your Loader object is already running under that name in this instance of Lookout*Direct*. Only one process may run under each process name.

Load: No state file specified

You must specify state file information for any process you load with a Loader object.

Load: Invalid process name

You have chosen an invalid process name. See the *Object Names* section in Chapter 4, *Using LookoutDirect*, of your *Getting Started With LookoutDirect* manual for detailed information on valid Lookout*Direct* process and object names.

Load: Invalid database computer name

The computer name you have entered in the **ComputerName** field is not registered as being a part of your Lookout*Direct* network. Check the spelling of the name or register the computer through the Lookout*Direct* Object Explorer or the Lookout*Direct* Connection Browser.

Load: Invalid database folder

Check the path you have entered in the **Citadel Database Folder** field to make sure the path you entered does exist on the computer specified in the **ComputerName** field.

Load: Can't open process file *processfilename*

Lookout*Direct* was unable to open the file `processfilename`. Check to see that the file exists in the specified location and has not been corrupted.

Unload: No process name specified

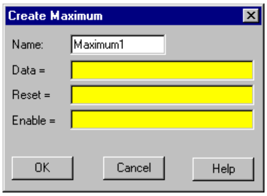
Enter the name of the process you want to unload in the **Process Name** field.

Unload: Process not found

Lookout*Direct* must be running a process with the name specified in the **Process Name** field for the Loader to unload a process. If no process with the specified name is running when the Loader attempts to unload a process, Lookout*Direct* returns this error message.

Maximum

Maximum actively calculates the maximum value of **Data** over time. Maximum is only active when the **Enable** expression is TRUE. It resets to zero when the **Reset** expression transitions from OFF to ON. Maximum also maintains an array of up to 35 previous maximum values. If **Enable** is left blank, the object always actively calculates the maximum. **Data** is a numeric expression while **Reset** and **Enable** are logical expressions.



Create Maximum

Name: Maximum1

Data =

Reset =

Enable =

Note Maximum does not have a display parameters dialog box. However, you can easily display Maximum by referencing it in an expression.

Maximum Data Members

Table 2-30. Maximum Data Members

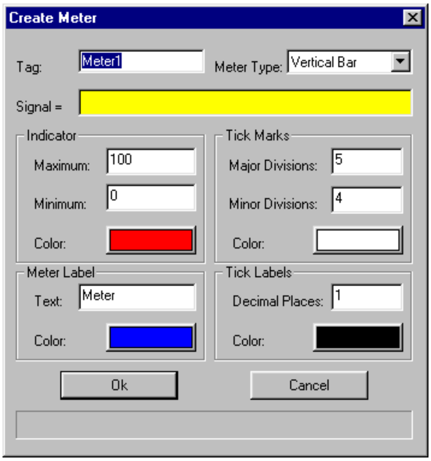
Data Members	Type	Read	Write	Description
(implicit)	numeric	yes	no	Current maximum value
1 – 35	numeric	yes	no	Previous maximum values. Signal 1 is the most recent prior maximum since the Reset expression went high.
DataReset	logical	no	yes	Upon transition from FALSE to TRUE, resets to zero all data members—including the current maximum value and all previous maximum values.

Comments The **Reset** interval could be a regular pulse interval created by a TimeOf*xxxx* timer, so that the pulse is synchronized to the top of the hour or day. For example, if you want to calculate the daily maximum flow rate, use the output signal from a TimeOfDay timer or a daily Spreadsheet object to reset the maximum calculation at the beginning of each day.

Related Objects *Average*, *Minimum*, *Sample*, *SampleText*

Meter

The Meter object displays the Signal expression in a vertical bar graph format or as a sweeping needle on a speedometer dial. Meter display values change depending on the derived numeric value of the Signal expression.



Create Meter

Tag: Meter1 Meter Type: Vertical Bar

Signal =

Indicator
Maximum: 100
Minimum: 0
Color: 🟥

Tick Marks
Major Divisions: 5
Minor Divisions: 4
Color: ⬜

Meter Label
Text: Meter
Color: 🟦

Tick Labels
Decimal Places: 1
Color: ⬛

Minimum is the lowest value signal the pot will generate.

Indicator parameters include Maximum and Minimum which define the span of the signal to be displayed. The Indicator Color is the color of the needle or vertical bar fill.

MeterLabel parameters allow for a Text label to be assigned during the meter configuration and the text Color to be selected.

Tick Marks options include parameters for the number of Major Divisions and the number of Minor Divisions along with a selection for the color of the divisions.

Note The Minor Divisions are the number of minor ticks between each two major divisions. Thus 5 Major Divisions and 4 Minor Divisions will result in whole number divisions, similar to 11 Major Divisions and 4 Minor Divisions.

Tick Labels options include parameters for the number of Decimal Places to be displayed on the numeric label of the Major Divisions along with a selection for the color of the numeric label.

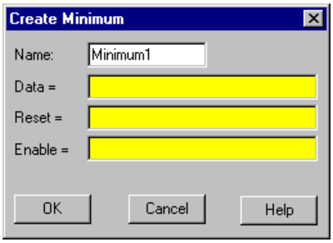
Meter Data Members

Table 2-31. Maximum Data Members

Data Members	Type	Read	Write	Description
(implicit)	numeric	yes	no	Current value of signal parameter

Minimum

Minimum actively calculates the minimum level of **Data** over time. Minimum is only active when the **Enable** expression is TRUE. It resets to zero when the **Reset** expression transitions from OFF to ON. Minimum also maintains an array of up to 35 previous minimum values. If **Enable** is left blank, the object is always actively calculating the minimum. **Data** is a numeric expression while **Reset** and **Enable** are logical expressions.



Name: Minimum1

Note Minimum does not have a display parameters dialog box. You can easily display the Minimum value referencing it an expression.

Minimum Data Members

Table 2-32. Minimum Data Members

Data Members	Type	Read	Write	Description
(implicit)	numeric	yes	no	Current minimum value
1 – 35	numeric	yes	no	Previous minimum values. Signal 1 is the most recent prior minimum since the Reset expression went high.
DataReset	logical	no	yes	Upon transition from FALSE to TRUE, resets to zero all data members—including the current minimum value and all previous minimum values.

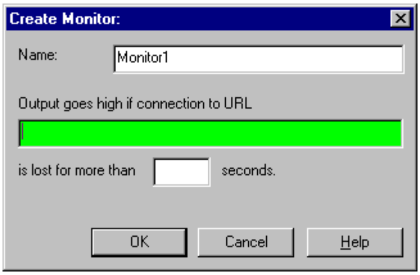
Comments The **Reset** interval could be a regular pulse interval created by a TimeOf*xxxx* timer, so that the pulse is synchronized to the top of the hour or day. For example, if you want to calculate the daily minimum flow rate, use the output signal from a TimeOfDay timer or a daily Spreadsheet object to reset the minimum calculation at the beginning of each day.

Related Objects *Average*, *Maximum*, *Sample*, *SampleText*

Monitor

The Lookout*Direct* Monitor object is integral to Lookout*Direct* redundancy. The Monitor object maintains a connection with a data member in any process you want to watch from another computer or process. If the data flow from that source stops, the Monitor signals the halt, allowing you to respond.

Using a Monitor object in conjunction with a Loader object is how you create failover redundancy with Lookout*Direct*. See Chapter 10, *Redundancy*, in the *LookoutDirect Developer's Manual* for more information on redundancy.


Create Monitor:

Name: Monitor1

Output goes high if connection to URL

is lost for more than seconds.

In the **Output goes high** field enter a data member in the process you want to monitor. If the data quality of that expression goes bad for more than the number you enter in the **seconds** field, the monitor goes high (TRUE), letting you know there is a problem with the process.

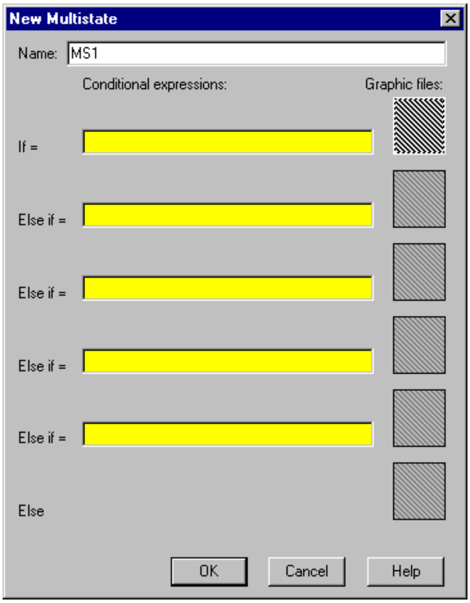
Monitor Data Members

Table 2-33. Monitor Data Members

Data Member	Type	Read	Write	Description
(implicit)	logical	yes	no	Goes TRUE when the data member the Monitor is watching cannot be accessed after a specified length of time.
timeout	numeric	yes	no	The length of time in seconds that the data member being monitored must be unavailable before the Monitor goes TRUE.

Multistate

Multistate displays different graphics on a control panel as dictated by the values of **Conditional expressions**. You can use up to six **Graphic files**, but at least one is required. Multistate determines which graphic to display based on the order and current status of your **Conditional expressions**. If several **Conditional expressions** are true at once, Multistate displays the graphic associated with the first true expression.



New Multistate

Name: MS1

Conditional expressions must result in logical values (TRUE or FALSE). See the *Animator* section for more information about constructing logical statements. Double-click in a **Graphic file** box to select the graphic you want to use.

Multistate Data Members

Table 2-34. Multistate Data Members

Data members	Type	Read	Write	Description
none	—	—	—	Multistate objects do not have data members

Comments By creating several graphic images that depict a sequence of events, Multistate can be used to create animation sequences on control panels such as hydraulic pistons moving back and forth. A more typical use of Multistate is for three-color pilot lights, where green represents running, red represents stopped, and yellow represents failed, for example.

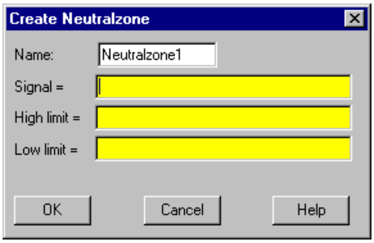
For smooth, high speed animations, use the Animator object.

Related Objects *Animator*

Neutralzone

Neutralzone is an ON/OFF Controller. It functions the way a home air conditioning thermostat does; if the temperature rises above a certain level, the `above` data member goes TRUE (turning the A/C on). When it drops below a lower temperature, the `below` data member goes TRUE (turning the A/C off).

When the incoming **Signal** value rises above both **Low limit** and **High limit**, the data member `above` turns on, and the data member `below` turns off. When the incoming **Signal** value drops below both **Low limit** and **High limit,** `above` turns off, and `below` turns on. The `above` and `below` data members do not change state when the signal value falls back within the two limits (within the neutral zone). **Signal**, **High limit**, and **Low limit** are all numeric expressions.


Create Neutralzone

Name: Neutralzone1

Signal =

High limit =

Low limit =

The previous discussion assumes numeric constants for both limits. However, you could use variable setpoint signals from Pot objects so an operator could dynamically adjust Neutralzone behavior.

Note Neutralzone does not have a display parameters dialog box. You can easily display the result of Neutralzone output signals by referencing its data members in an expression.

Neutralzone Data Members

Table 2-35. Neutralzone Data Members

Data Members	Type	Read	Write	Description
above	logical	yes	no	ON if signal is greater than both limits, OFF if signal is less than both limits, and does not change if signal is between both limits
below	logical	yes	no	ON if signal is less than both limits, OFF if signal is greater than both limits, and does not change if signal is between both limits

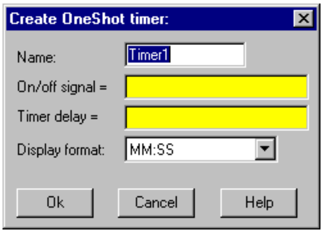
Comments You can use this object to turn pumps on and off or open and close valves based on line pressures or tank levels. Neutralzone objects prevent pumps from cycling on and off around a single setpoint, just as an air conditioning thermostat prevents your home air conditioner from incessantly starting and stopping.

Often the term deadband is mistakenly used to describe a neutral zone. However, deadbands refer to the amount of change a numeric value must travel in the reverse direction before the output numeric value begins to change.

OneShot

OneShot is an adjustable delay timer. When **On/off signal** transitions to on, the output signal goes TRUE and the **Timer delay** begins to count down. At the end of the delay countdown, the output signal goes FALSE.

Unlike the Interval timer, the OneShot timer output remains on for the **Time delay** period even if **On/off signal** goes FALSE. So a OneShot timer requires only a momentary signal to begin the **Timer delay** period. Pulsing the **On/off signal** during the time delay period does not extend the time delay period of a OneShot timer.



Create OneShot timer:

Name: Timer1

On/off signal =

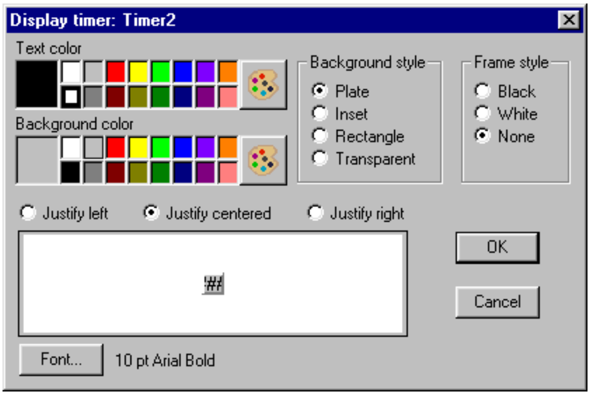
Timer delay =

Display format: MM:SS

The **On/off signal** is a logical expression while Timer delay is a numeric expression. Normally, this is a simple time constant such as 0:20 (twenty seconds). See *Numeric Data Members* in Chapter 2, *How LookoutDirect Works*, of the *Getting Started with LookoutDirect* manual for information on time constants.

Timer delay can range from 0.0 seconds to several years. The effective resolution is 0.1 seconds over the entire range.

The object is represented on a control panel by showing the time delay remaining in the format defined by the **Display format** parameter. It is updated approximately once per second. If the delay period has expired, it shows **`OFF`**.



Display timer: Timer2

Text color

Background color

Background style: ● Plate ○ Inset ○ Rectangle ○ Transparent

Frame style: ○ Black ○ White ● None

○ Justify left ● Justify centered ○ Justify right

##

Font... 10 pt Arial Bold

OneShot Data Members

Table 2-36. OneShot Data Members

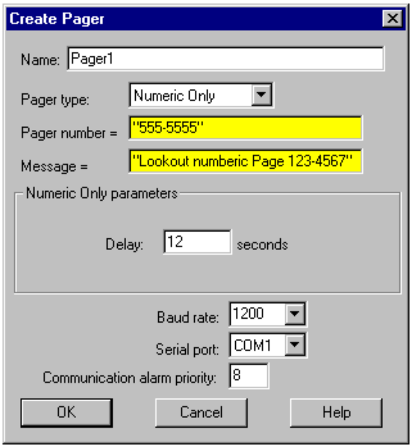
Data Member	Type	Read	Write	Description
(implicit)	logical	yes	no	Logical timer value

Comments You can use the OneShot timer to hold a valve open for a set period of time after a pushbutton is pressed, or to prevent pump starts from occurring too rapidly in succession.

Related Objects *DelayOff*, *DelayOn*, *Interval*, *Pot*, *TextEntry*

Pager

Pager is an object class Lookout*Direct* uses to contact a numeric or alphanumeric pager through a modem, sending a message to the pager.



Name: Pager1

Pager type: Numeric Only

Pager number = "555-5555"

Message = "Lookout numberic Page 123-4567"

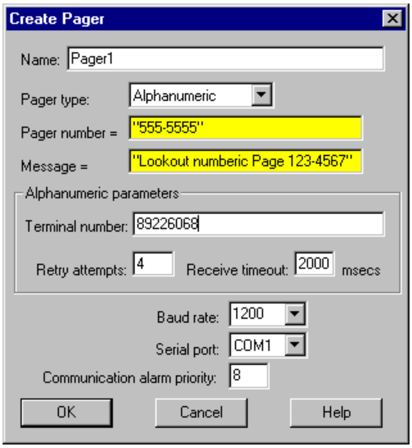
Numeric Only parameters

Delay: 12 seconds

Baud rate: 1200

Serial port: COM1

Communication alarm priority: 8



Name: Pager1

Pager type: Alphanumeric

Pager number = "555-5555"

Message = "Lookout numberic Page 123-4567"

Alphanumeric parameters

Terminal number: 89226068

Retry attempts: 4 Receive timeout: 2000 msecs

Baud rate: 1200

Serial port: COM1

Communication alarm priority: 8

Pager type determines whether the Pager object operates in numeric only or alphanumeric mode. A detailed description of the operation of these two modes follows.

Pager number is the phone number of the pager you want to contact. When the Pager object is in **Alphanumeric** mode, this number corresponds to the pager ID number.

Message is the message you want to send to the pager. Notice that in **Numeric Only** mode only numeric characters are sent.

Delay is how long the Pager object waits after dialing the pager number before it dials the message number. This parameter is valid in **Numeric Only** mode only.

Terminal number is the phone number of the remote paging terminal you want to contact. This parameter is valid in **Alphanumeric** mode only.

Retry attempts specifies the consecutive number of times Lookout*Direct* attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the Pager object generates an alarm and releases the COM port. Refer to Chapter 3, *Serial Communications,* in the *LookoutDirect Developer's Manual* for more information. This parameter is valid in **Alphanumeric** mode only.

Receive timeout is the time delay Lookout*Direct* uses in waiting for a response from a device before retrying the request. This parameter is valid in **Alphanumeric** mode only.

Baud rate indicates the baud rate Lookout*Direct* uses to communicate with the modem and paging terminal.

Serial port specifies which COM port Lookout*Direct* uses to communicate with your modem. You must have this COM port configured as dial-up under **Options»Serial Ports**.

Communication alarm priority determines the priority level of alarms generated by the Pager object. You can relate such alarms to communications with the modem or with the remote paging terminal.

Pager Data Members

Table 2-37. Pager Data Members

Data Member	Type	Read	Write	Description
Message	text	no	yes	Pager message
Phone	text	no	yes	Individual pager phone number or Page ID number
Send	logical	no	yes	Sends the message on transition from FALSE to TRUE

Pager Object Modes

Numeric Only

In **Numeric Only** mode, the Pager object establishes a connection with your local modem. Once this connection has been established and the pager number dialed, the Pager object waits for the time specified by **Delay**, then dials the number that is the message. Because the `Message` data member is a text value, the Pager object in **Numeric Only** mode omits any non-numeric characters from the message when it is sent.

Alphanumeric Mode

In **Alphanumeric** mode, the Pager object actually establishes a connection with a remote paging terminal, then transmits an alphanumeric message using Telocator Alphanumeric Protocol (TAP) version 1.8. TAP is an industry standard protocol for paging terminals that accept alphanumeric pages. Alphanumeric messages are limited to 250 characters. The text value in the `Message` data member will be truncated to this length if it is longer.

Pager Serial Port Settings

Notice that there are two different retry settings that affect the operation of the Pager object in **Alphanumeric** mode. The retry settings in the Pager object dialog box govern serial communications with the remote paging terminal. This means that after the two modems have connected and finished handshaking, and the serial transaction is underway, each individual frame is timed by the **Receive timeout** setting, and retried the number of times specified by **Retry attempts**.

These retry settings will not dial the phone number again if the remote paging terminal for some reason does not answer or is busy, which happens occasionally. This setting and other important modem settings (including the AT initialization string that the Pager object must use on your modem) can be found in **Option»Serial Ports**, and should be chosen carefully. These settings are important in both Pager object modes.

Note You may have to increase your **Receive Gap** setting from its default of 5 to something closer to 20 or 25. You must also have your COM port configured as dial-up.

Pager Queueing

The Pager object queues up to 10 messages in either mode. If the object is in the process of sending out a page and the `Send` data member goes high again, the current value of the `Message` data member will be queued and sent out when the Pager object has time. Messages that are already in the queue will not be duplicated.

Pager Status Messages

Paging terminal refused logon

Alphanumeric only error code

Paging terminal forced disconnect

Alphanumeric only error code

Paging terminal NAKed block transmission

Alphanumeric only error code

Paging terminal abandoned block transmission

Alphanumeric only error code

No response within timeout period

This means that the modem is not responding to Lookout*Direct* requests.

Queue full

The paging queue currently has 10 pages in it, and will not accept any more until at least one of those pages is successfully sent.

Garbled response

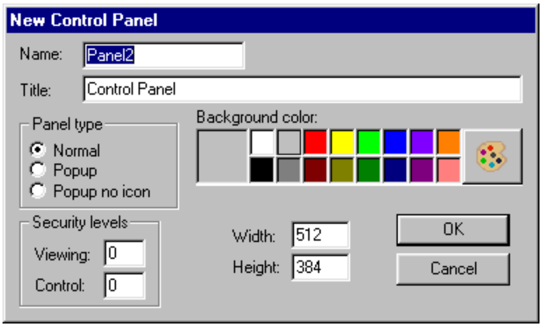
A response from the modem was corrupted or in an unrecognizable form.

Panel

Panel is a unique object class that accepts display members of other objects. Panels (Control panels) are a window into your process you can use to monitor and control your system by flipping switches, depressing pushbuttons, and turning knobs.

There is no limit as to the number of control panels you can create in Lookout*Direct* or the number of objects or graphics that you can display on a given panel. Control panels can be any size, and you can display them within the Lookout*Direct* workspace in various states (maximized, normal, minimized).

Create control panels with the **Object»Create…** command or with the **Insert»Control panel…** command. Both commands deliver the same result.



New Control Panel

Name: Panel2

Title: Control Panel

Panel type

● Normal

○ Popup

○ Popup no icon

Security levels

Viewing: 0

Control: 0

Width: 512

Height: 384

There are three distinct panel types: **Normal**, **Popup**, and **Popup no icon**.

A **Normal** control panel can be maximized, normal size, or minimized within the Lookout*Direct* workspace. When you activate a **Normal** panel it appears at the size defined by its **Height** and **Width**. When you maximize a **Normal** panel, it fills the Lookout*Direct* workspace. When you minimize a **Normal** panel it appears as an icon. The **Normal** option is typically selected for full-sized control panels.

Popup control panels can either be in a popup state or minimized (they cannot be maximized). When you activate a **Popup** panel it appears at the size defined by its **Height** and **Width**. When a Popup control panel is activated, it remains on top of all other panels until it is minimized. When you minimize a **Popup** panel it appears as a bar at the bottom of the Lookout*Direct* screen.

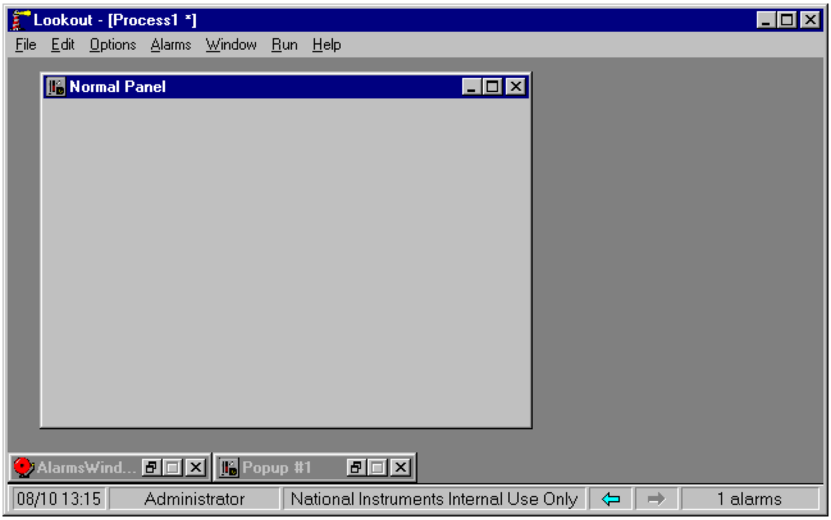
Popup no icon control panels are identical to **Popup** panels except they are not represented by icons when minimized. When you minimize a **Popup no icon** panel it disappears from the Lookout*Direct* workspace.

Note Because **Popup no icon** control panels are not represented by icons, they use less memory. (Microsoft Windows allocates a fixed amount of memory to each icon.) If you are experiencing memory problems when running a Lookout*Direct* application that has many control panels, consider converting your **Popup** panels to **Popup no icon**.

Normal panels and Popup panels can be chosen by selecting their icon, using the **Window** menu command. **Popup no icon** control panels cannot; they can only be accessed by triggering their `activate` data member.

The **Security levels** settings are globally applied to a given control panel. The **Control** security level works in conjunction with all individual object security levels on that panel. The higher security level of the two is used to determine if an operator has control over the object. For example, consider a single Switch object with a security level of 4 that is displayed on two panels. The first panel has a control level of 6 and the second panel has a control level of 2. Only level 6 and higher operators are able to flip the switch on the first panel; however, level 4 and higher operators have control over the same switch on the second panel.

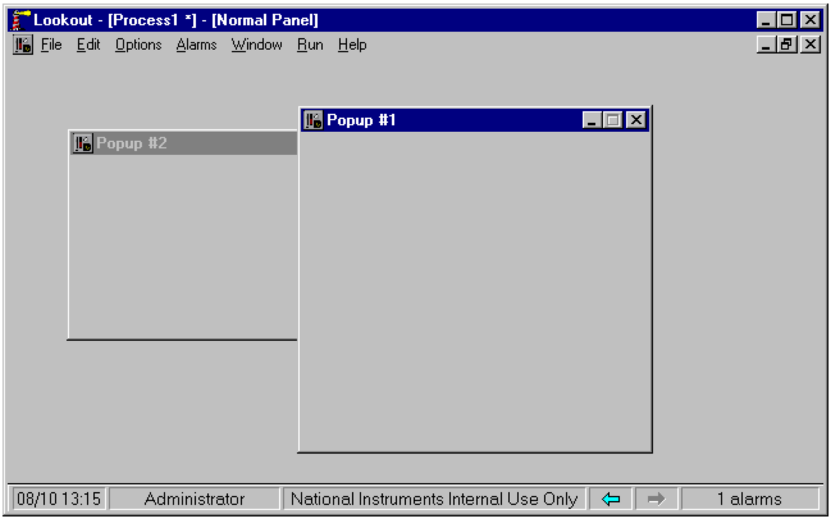
Viewing security can hide entire control panels from low level operators. This parameter affects the entire control panel. As an example take a control panel with a viewing security level of 6. If a level 5 (or lower) operator logs on, he is unable to see the control panel. In fact, he does not even know it exists because it is not listed in the **Window** menu and it is not shown as an icon. If a level 6 (or higher) operator logs on, the control panel instantly becomes available for display. This feature is useful for hiding panels that are rarely used or that contain sensitive information.



This example shows a **Normal** control panel in a normal state and a **Popup** control panel in a minimized state. The **Normal** panel could easily be minimized or maximized by hitting the appropriate arrow button.

The following example is a typical scenario involving full screen control panels with multiple popup panels displayed at once.

This configuration maximizes the amount of information that you can display at once; and it allows you to have any number of different combinations of control panels displayed on your monitor.


Lookout - [Process1 *] - [Normal Panel]
File Edit Options Alarms Window Run Help
Popup #2
Popup #1
08/10 13:15
Administrator
National Instruments Internal Use Only
1 alarms

This example displays a **Normal** control panel in a maximized state and two **Popup** control panels in a “popped up” state.

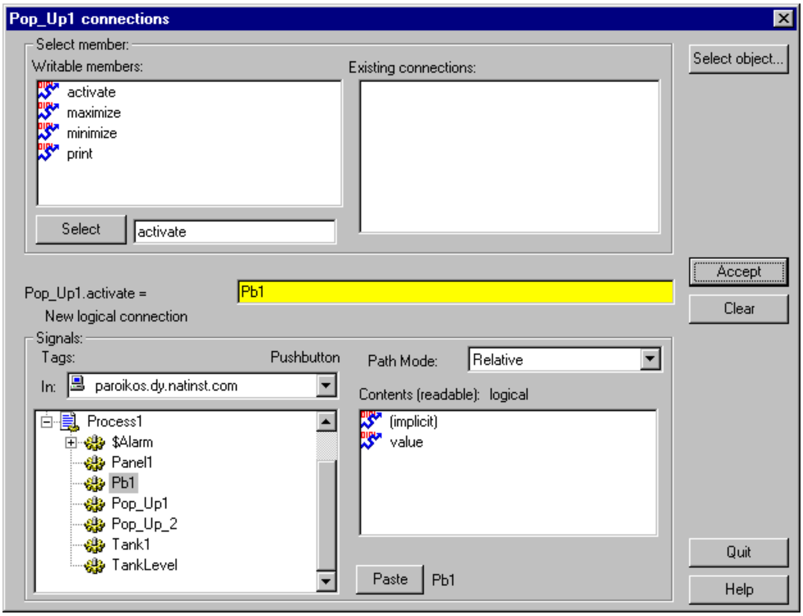
Manipulating Panels

Lookout*Direct* control panels utilize the Microsoft standard Multiple Document Interface (MDI) techniques. You manipulate Lookout*Direct* windows the same way you do other windows in the Microsoft Windows environment.

Panel Switching

It is common to have several or even dozens of control panels. Creating a methodology for moving between your panels can be as simple or as elaborate as you want. One effective method utilizes pushbuttons that invoke other control panels. You connect the pushbutton output signal to the activate or maximize data member of the control panel(s) you want to affect. When the button signal goes high the respective panel(s) appear.

The following example shows a single pushbutton and X with its output signal connected to two **Popup** control panels. The pushbutton is inserted on a third maximized panel.



Pop_Up1 connections

Select member:

Writable members:

- activate
- maximize
- minimize
- print

Existing connections:

Select: activate

Pop_Up1.activate = Pb1

New logical connection

Signals:

Tags: Pushbutton

In: paroikos.dy.natinst.com

- Process1
  - $Alarm
  - Panel1
  - Pb1
  - Pop_Up1
  - Pop_Up_2
  - Tank1
  - TankLevel

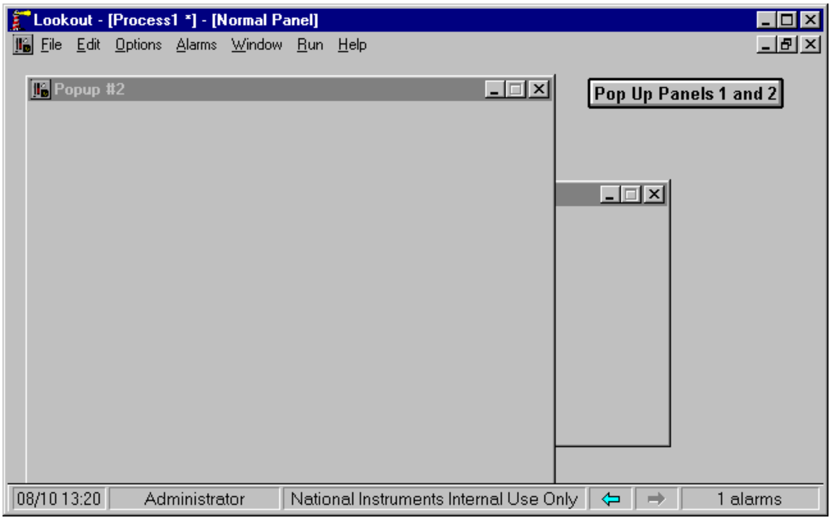
Path Mode: Relative

Contents (readable): logical

- (implicit)
- value

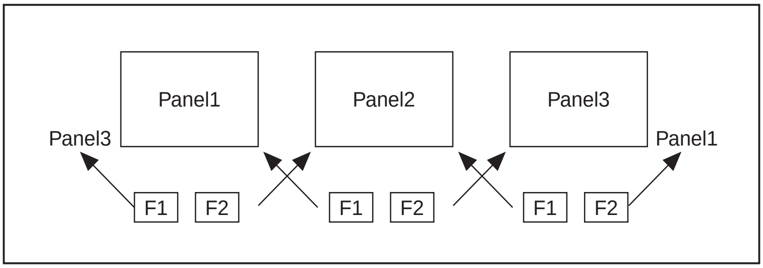
Paste: Pb1

When you toggle out of Edit Mode and depress the button, both **Popup** control panels instantly appear, as shown in the following illustration. Of course you could have connected the pushbutton to another **Normal** panel instead, and it would have appeared as the new maximized panel.


Lookout - [Process1 *] - [Normal Panel]
File Edit Options Alarms Window Run Help
Popup #2
Pop Up Panels 1 and 2
08/10 13:20
Administrator
National Instruments Internal Use Only
1 alarms

As you can imagine, there is no limit to the number of connections between various signals and control panels. In fact, you can create complex expressions/alarms that automatically call up specific control panels.

Another way to move between panels is through the use of function keys. Like the $Keyboard object, each panel has its own set of data members representing the function keys F1 – F24. The following example shows the F2 data member of Panel1 connected to the activate data member of Panel2. The F2 data member of Panel2 is also connected to the activate data member of Panel3. This way an operator can depress F2 to page forward through several panels.



In similar fashion, the F1 data member of Panel3 is connected to the activate data member of Panel2 and the F1 data member of Panel2 is connected to the activate data member of Panel1. So now, `Panel2.activate = Panel1.F2 OR Panel3.F1`. An operator can depress F2 to page forward through the control panels and F1 to page back through the control panels.

Special Considerations for "Home Panel"

To ensure users do not get lost when switching between panels, you might define one panel as your master control panel, or home panel or computer main menu. You could connect the activate data member of your home panel to $Keyboard.Shift.F1, or perhaps to a pushbutton object. If connected to the function key, any time the user presses <SHIFT-F1> (no matter what panel he or she is looking at), the home panel is called, returning the operator to a familiar control panel.

You might also want the home panel to maximize upon startup. If you have already created a pushbutton to call the home panel, you can connect it to the `maximize` data member.

The exclamation point (`!`) instructs Lookout*Direct* to use the opposite of the pushbutton value. At startup, the pushbutton is not depressed so its value is FALSE. But because you are using the opposite of the pushbutton value, Panel1.maximize is TRUE at startup. Any time a user depresses `CallHomePb` after this connection is made, nothing happens until the pushbutton is released—at which time the panel is called.

Panel Print

You can easily print a control panel using the **Print Panel** command. This function works equally well for both Normal and Popup panel types.

Print the contents of a panel by clicking on the panel control menu and then on **Print Panel**, or by connecting a logical expression to the `Print` data member of the desired control panel. A panel does not have to be visible to be printed.

Note Certain metafiles look different on the printed page than they do on the display screen. This means that parts of layered objects sometimes appear opaque on the screen, but translucent when drawn on paper.

Note When you print a **Normal** panel, it is printed at its defined **Height** and **Width** parameters. If you define a panel whose **Height** and **Width** are at the default

400 × 300 pixel setting, maximize the panel, and then add graphic elements to the full panel, those elements outside of the default 400 × 300 pixel range are not shown when the panel is printed. To print all the elements on a maximized panel, modify the **Width** and **Height** of the panel to match the full-screen dimension of the panel.

You can modify an existing control panel by toggling into edit mode and right-mouse clicking on its title bar. You can also modify a panel with the **Object»Modify…** menu command just like any other object.

Screen Resolution and Lookout*Direct* Graphics

Lookout*Direct* graphics and control panels appear different, depending on the PC you are using and the resolution of your screen driver. When you position a graphic (or any other display element) onto a control panel, Lookout*Direct* identifies the position you selected by recording the specific pixel position of the graphic. (A pixel is the smallest possible dot on the screen.) Lookout*Direct* actually counts the number of pixels that the graphic is from the upper left-hand corner of the screen. When you subsequently recall a panel, Lookout*Direct* knows the exact location to place the graphic.

The reason to bring this up is because different computer screen drivers have different screen resolutions. VGA screens are 640 × 480 pixels. Super VGA screens typically range from 800 × 600 pixels to 1024 × 768 pixels. A panel created at 640 × 480 pixel resolution does not fill the screen of a 1024 × 768 super VGA monitor. A panel created at 1024 × 768 pixel resolution will overflow the screen of a 640 × 480 VGA monitor.

It is best to create your panels using the display driver resolution of the computer on which you intend to run Lookout*Direct*. If you are creating panels for use on multiple computers, consider developing panels using the display driver resolution of the most common resolution monitor (if you have a dozen Super VGA computers and one VGA computer, develop your panels in Super VGA, not VGA). You will then have to modify the panels slightly to fit on the less common resolution computer(s).

You can usually change the resolution of your screen from VGA to Super VGA by changing System Settings in Windows Setup. Refer to your *Microsoft Windows* manual for more information.

Panel Data Members

Table 2-38. Panel Data Members

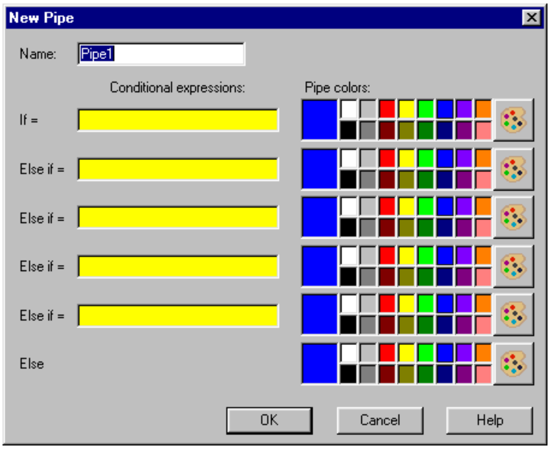
Data Members	Type	Read	Write	Description
<CTRL-F1> – <CTRL-F24>	logical	yes	no	Each of these 24 data members represent a function key, <F1> – <F24>—when pressed in conjunction with the <CTRL> key. Returns TRUE when the panel is active and its associated function key is pressed in conjunction with the <CTRL> key.
<F1> – <F24>	logical	yes	no	Each of these 24 data members represent a function key, <F1> – <F24>. Returns TRUE when the panel is active and its associated function key is pressed.
<SHIFT-F1> through <SHIFT-F24>	logical	yes	no	Each of these 24 data members represent a function key, <F1> – <F24>—when pressed in conjunction with the <SHIFT> key. Returns TRUE when the panel is active and its associated function key is pressed in conjunction with the <SHIFT> key.
activate	logical	no	yes	Upon transition from FALSE to TRUE, calls control panel to focus or pop up.
active	logical	yes	no	Returns TRUE when the panel is the currently selected panel (i.e., it is active).
maximize	logical	no	yes	Upon transition from FALSE to TRUE, maximizes control panel, replacing existing maximized control panel.

Table 2-38. Panel Data Members (Continued)

Data Members	Type	Read	Write	Description
minimize	logical	no	yes	Upon transition from FALSE to TRUE, minimizes control panel to icon state.
print	logical	no	yes	Upon transition from FALSE to TRUE, sends the control panel to the Windows Print Manager.

Pipe

Pipe displays different color rectangles (pipes) on a control panel as defined by the values of **Conditional expressions**. Pipe determines which color rectangle to display based on the order and current status of your **Conditional expressions**. If several **Conditional expressions** are TRUE at once, Pipe displays the color associated with the first TRUE expression. **Conditional expressions** must result in logical values.



New Pipe

Name: Pipe1

Pipe Data Members

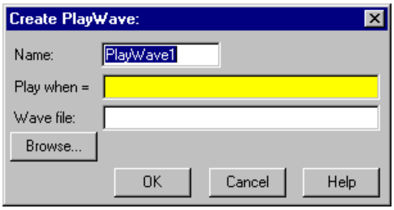
Table 2-39. Pipe Data Members

Data Members	Type	Read	Write	Description
none	—	—	—	Pipe does not have data members.

Comments You can easily create a complex piping network scheme (including changing colors) with a single Pipe object. Display the object on a control panel and copy the pipe display with the shift-drag method to create additional pipes with the same parameters. You can then move, resize, and group the pipes as you choose.

Playwave

Playwave connects Microsoft standard wave form files (`.wav`) to events in Lookout*Direct*. Playwave plays the audio file specified by **Wave file** when **Play when** transitions from off to on. **Play when** is a logical expression and might range from a simple pushbutton, to a digital input from a PLC, to an alarm generated in Lookout*Direct*. You can also create your own custom audio files with various software products. Therefore, you can connect individual alarms to custom wave files to be played each time an alarm goes TRUE.



Create PlayWave:

Name: PlayWave1

Play when =

Wave file:

Playwave Data Members

Table 2-40. Playwave Data Members

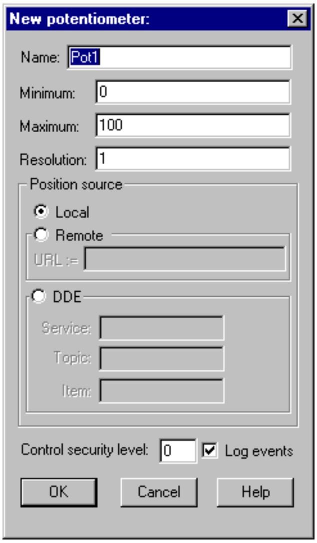
Data Members	Type	Read	Write	Description
none	—	—	—	Playwave does not have any data members

Comments Many computers do not come equipped with quality speakers built in. If this is the case, your wave files may sound distorted or may even be inaudible. If you want to take advantage of the Playwave feature, you may need to buy additional hardware, in particular a Microsoft Windows compatible sound board (with Windows driver) and external speakers.

Pot

Pot is a potentiometer that you use to change numeric setpoint values. You can display Pots on a control panel as a knob, vertical slider, horizontal slider, increment/decrement pushbuttons, or digital entry. You can also use Pots as multiple-position switches.

If you change the background color of a panel and add a Pot object displayed as a slider, its color is always gray. To change the background color of a Pot to match your panel, select the Pot object, then pick **Change»Background Color** from the menu.



New potentiometer:

Name: Pot1

Minimum: 0

Maximum: 100

Resolution: 1

Position source

● Local

○ Remote

URL :=

○ DDE

Service:

Topic:

Item:

Control security level: 0 ☑ Log events

Minimum is the lowest value signal the pot will generate.

Maximum is the highest value signal the pot will generate.

Resolution is the smallest increment of change, or detent spacing the Pot supports.

Position source determines where the value of the Pot resides. **Local** indicates the value of the Pot lies within the object itself—on the control panel.

Remote Pots get their values from a remote source, often the register on a controller they are connected to. Adjusting the Pot changes the value in the register, and changing the value in the register adjusts the Pot. In effect, the Pot is tracking a remote value. This is especially useful when you want to prevent Lookout*Direct* from changing the value of setpoints or registers upon initial startup, or reconnection of lost communication.

The **Remote** option calls for a URL to locate the data member you want to connect to. The URL field is green, and you cannot use a complex expression as a URL. If you need to use one RadioButton for several purposes, you can use a Symbolic Link to make a more complex connection than that possible with a URL.

You can right-click in the **URL** field and use the **URL Editor** dialog box to assemble the URL, in the same way you use the Lookout*Direct* expression editor.

A remote position source connection is completely reciprocal. A change in your Lookout*Direct* control changes the data member that control is remoted to. Any change in that data member also changes the control. It is not necessary, and is incorrect, to use the **Edit Connections** dialog box to connect a control object to its controlled data member.

Because the remote connection is reciprocal, you can only make such a connection to a data member that is either writable, or readable and writable.

When a process with remoted controls first opens, those controls take their initial values by following the URL to read the data members they are remoted to. The control will be covered by a red X to indicate that the remote connection is not functioning.

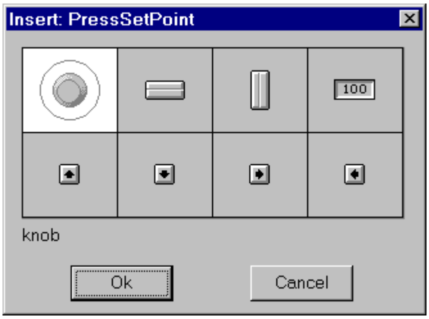
Note You should use **Remote** to connect a control in a client process to a data member in a server process. Connections inside a single process can be made using **Object»Edit Connections**.

Because complex expressions are read-only values, you cannot remote directly to them. For the same reason, you cannot remote one control to another control’s `(intrinsic)` data member (though you can remote a control to another control’s value data member).

Much like Remote Pots, **DDE** (Dynamic Data Exchange) Pots get their values from a remote source. This could be a cell in a spreadsheet, another DDE aware application, or a second copy of Lookout*Direct* running on the network. The last DDE parameters used on any object automatically become the default values for any new DDE object.

Control security level specifies the minimum security level operators must have to gain access to this individual object, and thus control it.

The **Log events** option creates a permanent audit trail for the object—who did what and when. All adjustments of the Pot are logged to disk, including the time of the adjustment, the operator account name, and what adjustment was made.



Note You can modify the background color on vertical and horizontal sliders with the **Change»Background color…** menu command. You can modify the font and font color of digital Pots using Change commands.

Pot Data Members

Table 2-41. Pot Data Members

Data Member	Type	Read	Write	Description
(implicit)	numeric	yes	no	Current value
decrement	logical	no	yes	When this data member value transitions from FALSE to TRUE, the implicit value of the Pot object decreases by the Pot **Resolution** amount.
enable	logical	no	yes	If TRUE (the default), enables DDE. If FALSE, disables DDE. The default value is on. This input is ignored for non-DDE TextEntry objects.

Table 2-41. Pot Data Members (Continued)

Data Member	Type	Read	Write	Description
enterValue	logical	no	yes	Under specific circumstances, when this value transitions from FALSE to TRUE, pops up a **Enter new value** dialog box for an operator to use in entering a value for the Pot. See the note following the table for more detailed information. The enterValue data member is designed for use under unusual circumstances, in particular when pointing devices are not available on a computer running Lookout*Direct*. Because of its unusual operation, this data member should not be used unless it is necessary for hardware reasons.
increment	logical	no	yes	When this data member value transitions from FALSE to TRUE, the implicit value of the pot object increases by the **Resolution** amount.
reset	logical	no	yes	While this value equals TRUE, the control will be set to the value in `resetvalue`.
resetvalue	numeric	no	yes	Sets the value a control will take when the reset data member transitions from FALSE to TRUE.
value	numeric	yes	yes	The current value of the control. If you have remoted this control, then `value` is the current value of the position source.
visible	logical	no	yes	When FALSE, the Pot object cannot be seen on the display panel. When TRUE, the Pot can be seen and controlled.

Note When the **`enterValue`** input transitions from **`FALSE`** to **`TRUE`**, and if the Pot is visible, if Lookout*Direct* is not in edit mode, and if the Pot has at least one digital display, the **Enter new value** dialog box pops up so an operator can input a value, just as if the operator had clicked on the digital display.

The numeric format and position used for the dialog box are based on the digital display for a Pot. Even if the panel containing the Pot digital display is inactive, the **Enter new value** dialog box will pop up. You can prevent this by predicating the **`enterValue`** input on the panel's **`active`** data member.

Comments Potentiometers are one of the most common control objects used in process controls. Using Pots, a plant operator can make setpoint changes with the mouse. Pots also work well as H-O-A switches. To create an HOA switch with a Pot, specify the minimum as 1, the maximum as 3, and the resolution as 1.

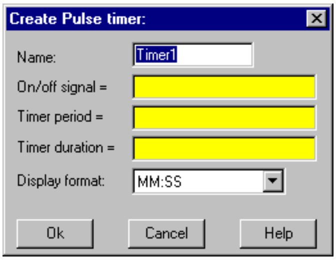
The increment and decrement data members enable quick connection of Pot objects to $Keyboard and Panel function keys, and screen Pushbuttons. These are often used to control Pot objects when Lookout*Direct* is running on an industrial PC platform that has restricted or no mouse functionality.

Pulse

Pulse is a timer that generates a periodic pulse of a specified duration. When **On/off signal** transitions to ON, the output signal turns on for the pulse duration time and then turns off for the remainder of the period. The output signal immediately turns off when the **On/Off signal** goes low.

Timer period is the time interval for the full pulse cycle, and **Timer duration** is the width of each pulse. These parameters can range from 0.0 seconds to several years, with an effective resolution of 0.1 seconds over the entire range. **Timer duration** should always be less than **Timer period**.

The object is represented on a control panel by showing the time remaining before the output changes state. It is depicted in the format defined by the **Display format** parameter. It is updated approximately once per second. If the **On/Off signal** is FALSE, it shows `OFF`.



Create Pulse timer:

Name: Timer1

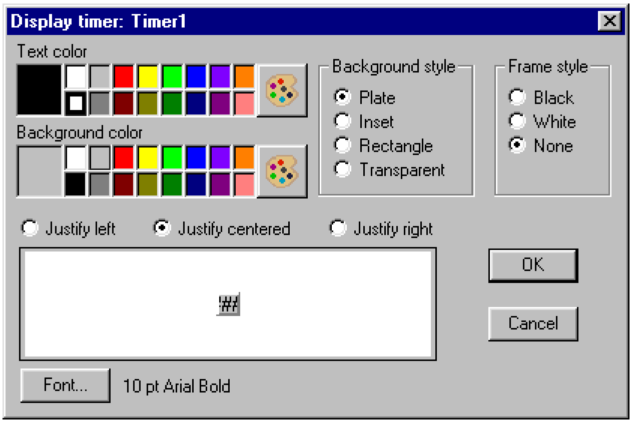
On/off signal =

Timer period =

Timer duration =

Display format: MM:SS

The **On/off signal** is a logical expression while **Timer period** and **Timer duration** are numeric expressions. Normally, these are simple time constants such as 0:20 (twenty seconds). See *Numeric Data Members* in Chapter 2, *How* Lookout*Direct Works*, of the *Getting Started with* Lookout*Direct* manual for information on entering time constants.



Display timer: Timer1

Text color

Background color

Background style
● Plate
○ Inset
○ Rectangle
○ Transparent

Frame style
○ Black
○ White
● None

○ Justify left ● Justify centered ○ Justify right

Font... 10 pt Arial Bold

Pulse Data Members

Table 2-42. Pulse Data Members

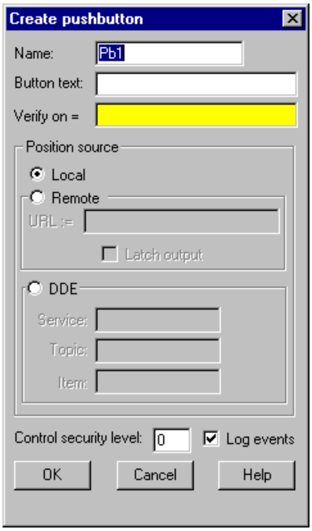
Data Members	Type	Read	Write	Description
(implicit)	logical	yes	no	Logical timer value

Comments Pulse can be used to periodically open a valve for a specified time duration. It can also act as a flasher to turn text and graphic signals on and off for display purposes.

Related Objects *DelayOff*, *DelayOn*, *Interval*, *OneShot*, *TextEntry*

Pushbutton

Pushbutton generates a logical signal for receipt by other objects. A pushbutton changes state when you position the cursor over it and press the mouse button, trackball, touchscreen, or space bar. The pushbutton remains depressed and the output signal remains high until you release the button. If a **Verify on** message is defined, the operator must first acknowledge the message, then the output signal goes high, *but only momentarily*.



Name: Pb1

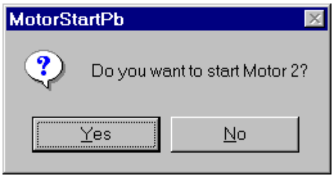
Position source: ● Local

Control security level: 0

☑ Log events

Button text displays the specified text on the pushbutton.

Use **Verify on** to create a dynamic text expression to be displayed in a message dialog box. See Chapter 6, *Security*, in the Lookout*Direct Developer's Manual* for more information on security.



Do you want to start Motor 2?

Position source determines where the value of the Pushbutton resides. **Local** indicates the value of the Pushbutton lies within the Pushbutton itself—on the control panel. If the pushbutton is not depressed its signal is OFF, if depressed its signal is ON.

Remote Pushbuttons get their values from a remote source, often the register in a controller they are connected to. Depressing the pushbutton changes the status of the register, and changing the status of the register depresses the pushbutton.

The **Remote** option calls for a URL to locate the data member you want to connect to. The URL field is green, and you cannot use a complex expression as a URL. If you need to use one RadioButton for several purposes, you can use a Symbolic Link to make a more complex connection than that possible with a URL.

You can right-click in the **URL** field and use the **URL Editor** dialog box to assemble the URL, in the same way you use the Lookout*Direct* expression editor.

A remote position source connection is completely reciprocal. A change in your Lookout*Direct* control changes the data member that control is remoted to. Any change in that data member also changes the control. It is not necessary, and is incorrect, to use the **Edit Connections** dialog box to connect a control object to its controlled data member.

Because the remote connection is reciprocal, you can only make such a connection to a data member that is either writable, or readable and writable.

When a process with remoted controls first opens, those controls take their initial values by following the URL to read the data members they are remoted to. The control will be covered by a red X to indicate that the remote connection is not functioning.

Note You should use **Remote** to connect a control in a client process to a data member in a server process. Connections inside a single process can be made using **Object»Edit Connections**.

Because complex expressions are read-only values, you cannot remote directly to them. For the same reason, you cannot remote one control to another control's `(intrinsic)` data member (though you can remote a control to another control's value data member).

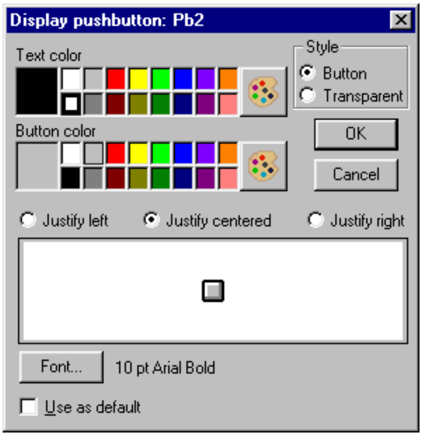
When you select the **Remote** option, you can choose whether or not the Pushbutton latches its output. The **Latch output** check box configures Lookout*Direct* for controlling a latching-relay.

When a user clicks on a Pushbutton that has latching selected, the pushbutton remains depressed, sending an ON signal (TRUE or high) until the Remote Position signal turns ON. Assume for example that an operator clicks on `MotorStartPb`, configured above. The pushbutton remains pushed in, sending a TRUE signal, until PLC.C101 goes TRUE. As soon as PLC.C101 goes TRUE, the pushbutton releases.

Much like Remote Pushbuttons, **DDE** (Dynamic Data Exchange) Pushbuttons get their values from a remote source. This could be a cell in a spreadsheet, another DDE aware application, or a second copy of Lookout*Direct* running on the network. See Chapter 5, *Dynamic Data Exchange,* in the Lookout*Direct Developer's Manual* for information on **Service**, **Topic**, and **Item** parameters.

Control security level specifies the minimum security level operators must have to gain access to this individual object, and thus control it. See Chapter 6, *Security*, in the Lookout*Direct Developer's Manual* for more information on security.

The **Log events** option creates a permanent audit trail for the object—who did what and when. Any depression of the Pushbutton is recorded to disk, including the time the button was depressed, and the operator's account name. See Chapter 7, *Logging Data and Events*, in the Lookout*Direct Developer's Manual* for more information on logging events.



Display pushbutton: Pb2

Text color

Button color

Style
● Button
○ Transparent

○ Justify left ● Justify centered ○ Justify right

Font... 10 pt Arial Bold

□ Use as default

Pushbutton Data Members

Table 2-43. Pushbutton Data Members

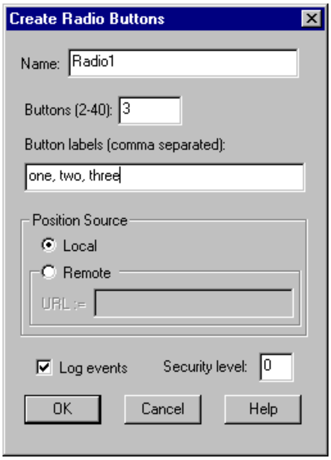
Data Members	Type	Read	Write	Description
(implicit)	logical	yes	no	Value of object (TRUE when button is depressed)
enable	logical	no	yes	If TRUE (the default), enables DDE. If FALSE, disables DDE. The default value is on. This input is ignored for non-DDE TextEntry objects.
reset	logical	no	yes	While this value equals TRUE, the control will be set to the value in `resetvalue`.
resetvalue	numeric	no	yes	Sets the value a control will take when the reset data member transitions from FALSE to TRUE.

Table 2-43. Pushbutton Data Members (Continued)

Data Members	Type	Read	Write	Description
value	numeric	yes	yes	The current value of the control. If you have remoted this control, then `value` is the current value of the position source.
visible	logical	no	yes	When FALSE, the Pushbutton cannot be seen on the display panel. When TRUE, the button can be seen and controlled.

RadioButton

The RadioButton object creates a set of radio buttons on your panel. You can have from 2 to 40 buttons in a set. Only one button at a time can be activated in each radio button group.



Create Radio Buttons

Name: Radio1

Buttons (2-40): 3

Button labels (comma separated): one, two, three

Position Source

● Local

○ Remote

URL :=

☑ Log events

Security level: 0

Set the **Buttons (2-40)** in each set. You can have as many as 40 buttons in each group, but you must have at least 2.

Position source determines where the value of the RadioButton resides. **Local** indicates the value of the RadioButton lies within the object itself—on the control panel.

The **Remote** option tells Lookout*Direct* to initialize the RadioButton value from a remote source. Adjusting the RadioButton changes the value in the register, and changing the value in the register adjusts the RadioButton. In effect, the RadioButton is tracking a remote value. This is especially useful when you want to prevent Lookout*Direct* from changing the value of setpoints or registers upon initial startup, or reconnection of lost communication.

The **Remote** option calls for a URL to locate the data member you want to connect to. The URL field is green, and you cannot use a complex expression as a URL. If you need to use one RadioButton for several purposes, you can use a Symbolic Link to make a more complex connection than that possible with a URL.

You can right-click in the **URL** field and use the **URL Editor** dialog box to assemble the URL, in the same way you use the Lookout*Direct* expression editor.

A remote position source connection is completely reciprocal. A change in your Lookout*Direct* control changes the data member that control is remoted to. Any change in that data member also changes the control. It is not necessary, and is incorrect, to use the **Edit Connections** dialog box to connect a control object to its controlled data member.

Because the remote connection is reciprocal, you can only make such a connection to a data member that is either writable, or readable and writable.

When a process with remoted controls first opens, those controls take their initial values by following the URL to read the data members they are remoted to. The control will be covered by a red X to indicate that the remote connection is not functioning.

Note You should use **Remote** to connect a control in a client process to a data member in a server process. Connections inside a single process can be made using **Object»Edit Connections**.

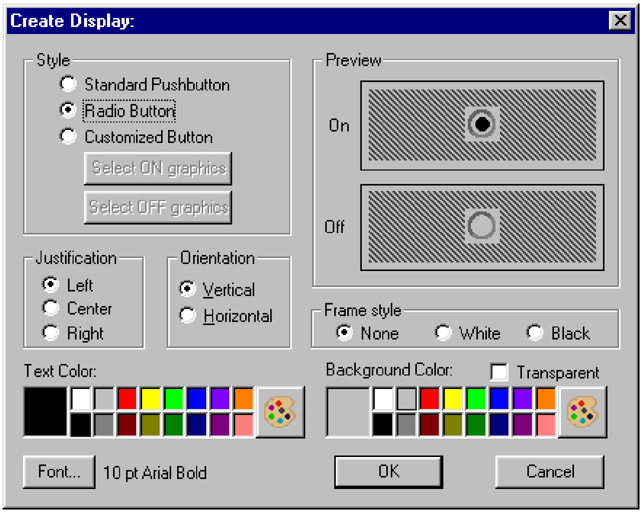
Because complex expressions are read-only values, you cannot remote directly to them. For the same reason, you cannot remote one control to another control's `(intrinsic)` data member (though you can remote a control to another control's value data member).

Note You cannot break a group of radio buttons into multiple rows.

Enter any text for the buttons in the **Button labels** field. Separate each button label with a comma. If you need to include a comma or a backslash in any button text, precede that character with a backslash(\). If you want to leave a button blank, you can enter two commas without any text or numbers between the commas. You do not have to enter a label for every button. Lookout*Direct* will leave each button after your last label entry blank. If you enter more labels than you have set buttons, your labels will be preserved but not displayed.

To log changes in button state, check the **Log Events** checkbox.

Security level sets the minimum security level an operator must have to change the radio buttons.



Create Display:

Style
○ Standard Pushbutton
● Radio Button
○ Customized Button

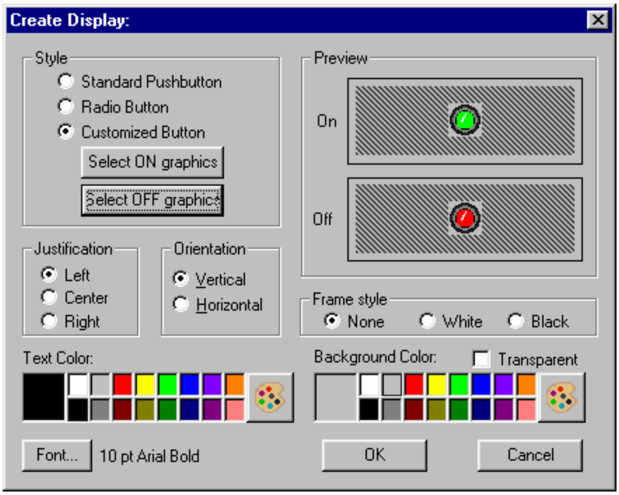
Justification: ● Left ○ Center ○ Right

Orientation: ● Vertical ○ Horizontal

Frame style: ● None ○ White ○ Black

Background Color: □ Transparent

Font... 10 pt Arial Bold



Create Display:

Style
○ Standard Pushbutton
○ Radio Button
● Customized Button

Justification: ● Left ○ Center ○ Right

Orientation: ● Vertical ○ Horizontal

Frame style: ● None ○ White ○ Black

Background Color: □ Transparent

Font... 10 pt Arial Bold

You can use a standard Lookout*Direct* pushbutton, standardized radio buttons, or custom graphics for your radio button groups.

Notice that you cannot enter text in the **Create Display** dialog box. You must enter any text you intend to use in the **RadioButton Definition Parameters** dialog box. You can change font and text size in the **Create**

Display dialog box. The first text label you entered for your radio button is displayed in the **Preview** field so you see the effect of font changes.

The default display uses a standard **Radio Button** display. You can adjust text and background color for any of your display choices.

Selecting **Standard Pushbutton** displays standard on/off buttons in your radio button array.

Select **Customized Button** for a free range of choices from the Lookout*Direct* graphics library, or create your own ON/OFF indicators for a radio button array. See the *Creating Custom Graphics* section of Chapter 2, *Graphics*, in the Lookout*Direct Developer's Manual* for detailed information on creating your own graphics for use on a Lookout*Direct* control panel.

Set the **Orientation** of the radio button array to horizontal or vertical by checking the appropriate checkbox. Set label text **Justification** and button **Frame style** in the appropriate fields.

Related Objects *L3Pot*, *L3Pushbutton*, *L3Switch*, *Pot*, *Pushbutton*, *Switch*

RadioButton Data Members

Table 2-44. RadioButton Data Members

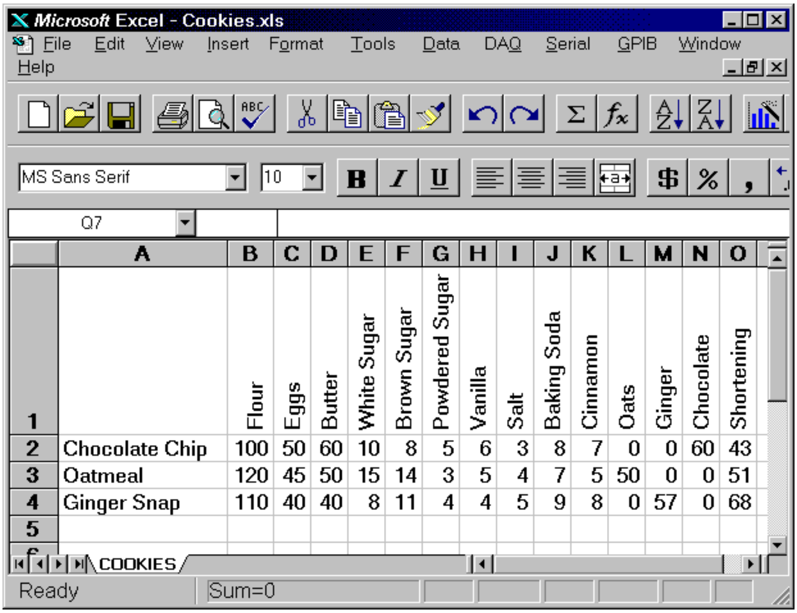
Data Member	Type	Read	Write	Description
1-40	logical	yes	no	Reports ON/OFF status of each button in a radio button set.
current	numeric	yes	no	Reports which radio button is active in a set.
(implicit)	numeric	yes	yes	Radiobutton value.
value	numeric	yes	yes	Number of the currectly active radiobutton.
visible	logical	yes	yes	Controls visibility of a radio button set. Default=TRUE.

Recipe

Recipe objects are an efficient means of importing large arrays of data (namely recipes and their ingredients) into Lookout*Direct* using an Excel (`.xls`) spreadsheet. Once created and implemented, the operator can easily and quickly change the current recipe with the click of the mouse, thus selecting a new set of ingredients.

The best way to describe how the Recipe class works is to step through a typical example, in this case involving cookie manufacture.

There are two steps to creating and implementing a recipe object. First, you define your recipes with their respective ingredients in a spreadsheet program such as Excel (anything that creates an `.xls` file will work—including Lotus 123). You can define up to 1,000 recipes in a single `.xls` file. Each recipe can have up to 255 ingredients. Three cookie recipes are defined in this spreadsheet.


Microsoft Excel - Cookies.xls
File Edit View Insert Format Tools Data DAQ Serial GPIB Window Help
MS Sans Serif
10
B I U
$ % ,
Q7
COOKIES
Ready
Sum=0

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		Flour	Eggs	Butter	White Sugar	Brown Sugar	Powdered Sugar	Vanilla	Salt	Baking Soda	Cinnamon	Oats	Ginger	Chocolate	Shortening
2	Chocolate Chip	100	50	60	10	8	5	6	3	8	7	0	0	60	43
3	Oatmeal	120	45	50	15	14	3	5	4	7	5	50	0	0	51
4	Ginger Snap	110	40	40	8	11	4	4	5	9	8	0	57	0	68
5															

The first row of the spreadsheet is reserved for ingredient names. They begin in column B. These ingredient names later become alias data members of the Recipe object. Therefore, ingredient names must be unique. They cannot have the same name as a native data member (see the Recipe Data Member table for data member names). Also, you cannot name an ingredient `missing`; this is a reserved word. Valid characters in an ingredient name include A – Z, 0 – 9, the dollar sign ($), and a period (.). If you enter `Hi_@#!!There` as an ingredient name, Lookout*Direct* names the alias `HiThere`. Alias names are case sensitive.

Beginning in Row 2, Column A lists the names of the various recipes. Recipes follow the same naming convention as ingredients. Each recipe is followed by its unique ingredient values.

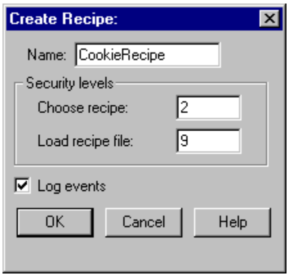
Ingredient values can represent process inputs, parameters, and outputs. Process inputs typically represent raw materials and other inputs consumed in the batch process. Examples include the number of eggs consumed, amount of flour used, amount of energy consumed, possibly even work hours required or amount of traceable fixed costs consumed.

Another type of ingredient value is a process parameter. Process parameters identify operational settings such as furnace cooling time, an air pressure setpoint, or a Low pH alarm limit. Process parameters might also include identifications of specific equipment to be used during the batch process.

The third type of ingredient value is a process output. Such an ingredient value might represent the number of finished cookies expected from the batch, amount of byproduct expected, or a cost variance calculation based on the selected recipe.

Ingredient value quantities may be specified as constants or as equations based on other formula parameters such as batch size.

The second part of defining a recipe involves defining a recipe object in Lookout*Direct*.


Create Recipe:

Name: CookieRecipe

Security levels

Choose recipe: 2

Load recipe file: 9

☑ Log events

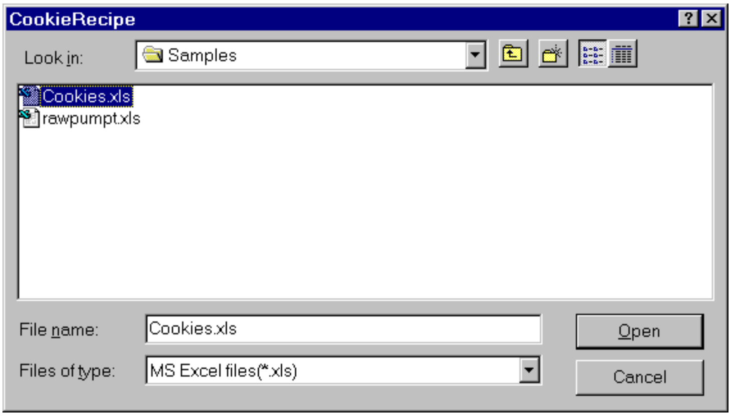
The first recipe dialog box defines security and event data logging.

Choose recipe security level specifies the minimum security level an operator must have to be able to select a recipe from all the recipes listed in the currently selected spreadsheet file.

Load recipe file security level specifies the minimum security level an operator must have to be able to select a different spreadsheet file.

The **Log events** option creates a permanent audit trail for the object—who did what and when. Any selection of a different recipe or recipe file is logged to disk, including the time the action occurred and the operator's account name. See Chapter 7, *Logging Data and Events*, in the Lookout*Direct Developer's Manual* for more information on logging events.

After defining security and event data logging, Lookout*Direct* presents you with a file selection dialog box.



Look in: Samples

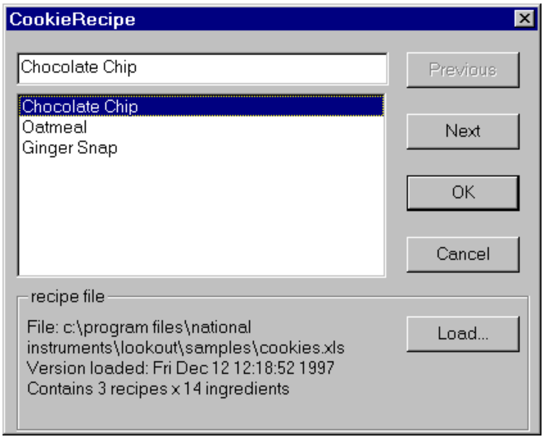
Cookies.xls
rawpumpt.xls

File name: Cookies.xls

Files of type: MS Excel files(*.xls)

Select the `cookies.xls` spreadsheet file because it has the three batch recipes in it.

Once a file is selected, Lookout*Direct* presents a list of recipes.



CookieRecipe

Chocolate Chip

Chocolate Chip
Oatmeal
Ginger Snap

recipe file

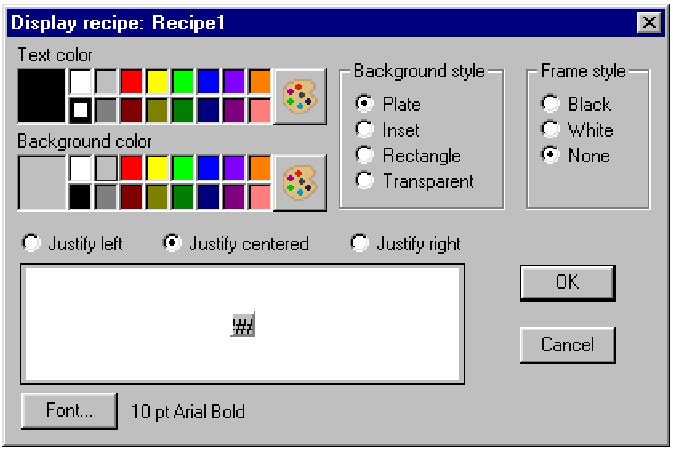
File: c:\program files\national instruments\lookout\samples\cookies.xls
Version loaded: Fri Dec 12 12:18:52 1997
Contains 3 recipes x 14 ingredients

As you can see, the recipe names come from column A of the spreadsheet. You can use the **Previous** and **Next** buttons to identify a recipe from among the list, or type the name in the data entry field above the list. (This same dialog box appears later when an operator clicks on the pushbutton representing this object.) Click on **OK** to choose the recipe you want to use. Select the recipe for oatmeal cookies.

The recipe file **Load** button invokes the file list dialog box, described previously. Click on this button to select a new `.xls` file.

Note If the recipe is changed in the spreadsheet, the change is noted in the recipe file dialog box—but the values currently resident in the object data members remain intact. The operator must load the spreadsheet again to update the copy of the recipe file in Lookout*Direct*. If you select a new `.xls` file to load but click on the Cancel button, it does *not* update! You must select **OK** for the recipe to actually be loaded.

After loading the spreadsheet, Lookout*Direct* presents the display parameters box.



Display recipe: Recipe1

Text color

Background color

Background style
● Plate
○ Inset
○ Rectangle
○ Transparent

Frame style
○ Black
○ White
● None

○ Justify left ● Justify centered ○ Justify right

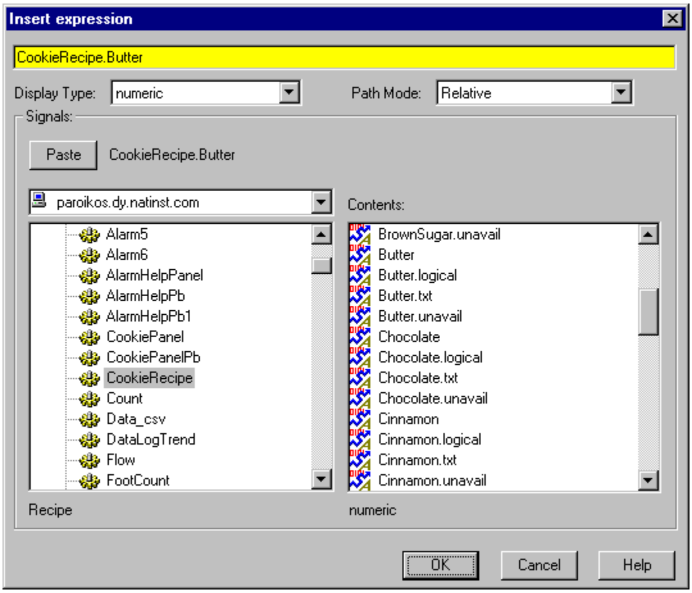
##

Font... 10 pt Arial Bold

After you choose the object display parameters, you can paste it into the panel.

When you select a new recipe, Lookout*Direct* writes the ingredient values for the selected recipe into the corresponding data members of the object.

The actual number of data members that a recipe object has is based on the number of ingredients within it. This is best demonstrated by looking at the Insert Expression dialog box.



CookieRecipe.Butter

Display Type: numeric

Path Mode: Relative

Signals:

CookieRecipe.Butter

paroikos.dy.natinst.com

Alarm5
Alarm6
AlarmHelpPanel
AlarmHelpPb
AlarmHelpPb1
CookiePanel
CookiePanelPb
CookieRecipe
Count
Data_csv
DataLogTrend
Flow
FootCount

Contents:

BrownSugar.unavail
Butter
Butter.logical
Butter.txt
Butter.unavail
Chocolate
Chocolate.logical
Chocolate.txt
Chocolate.unavail
Cinnamon
Cinnamon.logical
Cinnamon.txt
Cinnamon.unavail

Recipe

numeric

Notice that there are four data members for each defined ingredient. Actual data member names vary from object to object, depending upon your recipe ingredients. However, the four readable data member types for each ingredient are consistent.

Recipe Data Members

Table 2-45. Recipe Data Members

Data Member	Type	Read	Write	Description
(implicit)	text	yes	no	Name of currently selected recipe
B – IV	numeric	yes	no	Each letter, B through IV, represents a column in the spreadsheet. The value of the data member is the numeric amount of the ingredient for the currently selected recipe.

Table 2-45. Recipe Data Members (Continued)

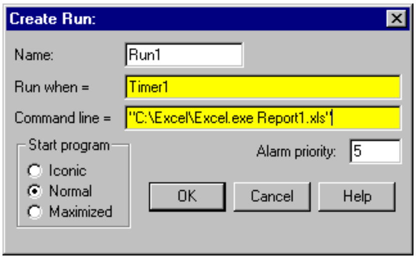
Data Member	Type	Read	Write	Description
B.logical through IV.logical	logical	yes	no	Each letter, B through IV represents a column in the spreadsheet. Returns TRUE (ON) if the amount of the ingredient in the spreadsheet cell for the selected recipe is greater than zero. Returns FALSE if the specified amount for the ingredient is zero.
B.txt – IV.txt	text	yes	no	Each letter, B through IV represents a column in the spreadsheet. The value of the data member is the textual amount of the ingredient for the currently selected recipe.
B.unavail through IV.unavail	logical	yes	no	Each letter, B through IV represents a column in the spreadsheet. Returns TRUE if the spreadsheet cell is empty. Returns FALSE if the cell contains data.
pick1 through pick1000	logical	no	yes	Upon transition from FALSE to TRUE, chooses the respective recipe within the spreadsheet. When used with pushbutton objects, these data members can eliminate the need for operators to use the recipe list dialog box.

Comments The recipe object reads a block of continuous columns. Therefore, the ingredient names should be a contiguous list in Row 1. If a recipe does not use a particular ingredient, just leave the respective cell blank.

When Lookout*Direct* encounters a blank cell in Column A, it ignores the entire row. Thus, you can easily annotate your recipes by leaving a cell in Column A blank and adding text to the cell in Column B of the same row.

Run

You can use Run objects to start an external program file from within Lookout*Direct*. When the result of the **Run when** logical expression goes TRUE, the object executes the **Command line**.



Create Run:

Name: Run1

Run when = Timer1

Command line = "C:\Excel\Excel.exe Report1.xls"

Start program

○ Iconic

● Normal

○ Maximized

Alarm priority: 5

In this example, Lookout*Direct* runs an Excel macro called `REPORT1.XLM` when the logical value returned by `Timer1` goes TRUE. `Timer1` is a TimeofDay object that triggers the report to run every day at 8:00 a.m.

The **Command line** text expression must be enclosed in quotation marks as shown. Notice that the example includes the full path name of the executable file. Ensure that your command line meets DOS syntax requirements. Because this is an expression data field, the command could be the result of a text expression.

Passing arguments in the Run Object in Lookout*Direct* requires proper use of double quotes depending on how many arguments you are passing and whether or not those arguments have spaces in the path.

If your argument has no spaces in the path, you need only insert double quotes before and after the argument.

If your argument has spaces in the path, you must insert two double quotes at the beginning and the end of that argument to define it as a single argument. These two double quotes are in addition to the double quotes you must use in passing any argument.

For example, when passing a single argument the following examples demonstrate the correct use of double quotes (spaces are exaggerated for effect):

```
"""c:\my excel\autoexec.xls"""

"c:\excel\excel.exe"
```

When passing two or more arguments, you follow the same rules for each argument, as illustrated in the following examples:

```
"""c:\program  files\msoffice\excel\excel.exe"" c:\autoexec.xls"
"""c:\program  files\excel\excel.exe"" ""c:\my excel\autoexec.xls"""
"c:\excel\excel.exe c:\working\autoexec.xls"
```

You can specify how an application presents itself when you activate it with the **Start program** selections. If you select **Normal**, the application window appears in front of the open Lookout*Direct* window when it is activated. If you want to reduce the application to an icon each time you start it, select **Iconic**. If you select **Maximized**, the application window replaces the Lookout*Direct* window on the screen. (Lookout*Direct* is still running; you just cannot see it. Press <Alt+Tab> to switch between applications.)

Table 2-46. Run Data Members

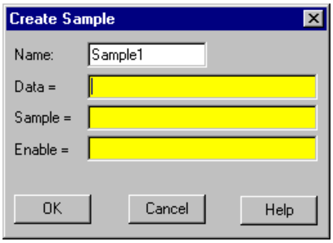
Data Member	Type	Read	Write	Description
none	—	—	—	Run does not have any data members

Comments If the application does not automatically shut down, multiple instances of the program may be running because of previous starts. Over time, this can snowball to the point where Windows performance is severely hampered.

If you want to execute DOS commands from within Lookout*Direct*, put the commands in a DOS batch file (`.bat`) and then identify the batch file in the Command Line.

Sample

The Sample object samples and holds data. Any time the **Sample** expression transitions from OFF to ON and the **Enable** expression is TRUE, the Sample object samples and stores a **Data** expression. Sample maintains an array of up to 35 previous samples. If **Enable** is left blank it is assumed to be TRUE. **Data** is a numeric expression while **Sample** and **Enable** are logical.



Name: Sample1

Data =

Sample =

Enable =

Note Sample does not have a display parameters dialog box. You can display the result of a Sample output signal by using its data member in an expression.

Sample Data Members

Table 2-47. Sample Data Members

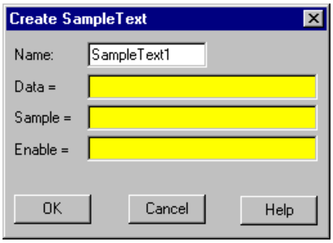
Data Member	Type	Read	Write	Description
(implicit)	numeric	yes	no	Current **Data** value—tracks **Data** input value.
1 – 35	numeric	yes	no	Previous samples. Signal 1 is the most recent sample since **Sample** went high. If you display Sample, you display the current and active value for the object—that is, the (implicit) value. If you display Sample.1, you display the least complete value.
DataReset	logical	no	yes	Upon transition from FALSE to TRUE, resets to zero all data members—including the current value and all previous samples.

Comments The **Reset** expression can be a regular pulse interval created by a TimeOf*xxxx* timer. For example, if you want to sample the temperature every hour of the day, use the output signal from a TimeOfHour timer in the **Reset** expression to sample the temperature at the beginning of each hour.

Related Objects *Average*, *Maximum*, *Minimum*, *SampleText*

SampleText

SampleText samples and stores the result of the **Data** expression any time the **Sample** expression transitions from OFF to ON and the **Enable** expression is TRUE. SampleText maintains an array of up to 35 previous samples. If **Enable** is left blank it is assumed to be TRUE.



Data is a text expression while **Sample** and **Enable** are logical expressions.

Note SampleText does not have a display parameters dialog box. You can display the result of a Sample object output signal by using its data member in an expression.

SampleText Data Members

Table 2-48. SampleText Data Members

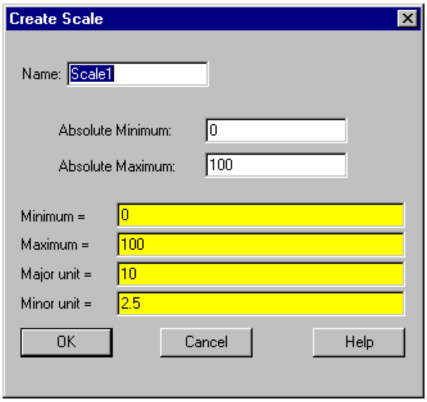
Data Member	Type	Read	Write	Description
(implicit)	text	yes	no	Current **Data** value. Tracks **Data** input value.
1 – 35	text	yes	no	Previous samples. Signal 1 is the most recent sample since **Sample** went high.
DataReset	logical	no	yes	Upon transition from FALSE to TRUE, resets all data members—including the current value and all previous samples.

Related Objects *Sample*, *L3TextEntry*, *TextEntry*

Scale

You can use the Scale object to create dynamic scales—that is, scales whose ranges and divisions can change based on numeric parameter expressions.

Note If you want to create a simple scale that does not change dynamically (which is normally the case), use the **Insert»Scale** command.



Name: Scale1

Absolute Minimum: 0

Absolute Maximum: 100

Minimum = 0

Maximum = 100

Major unit = 10

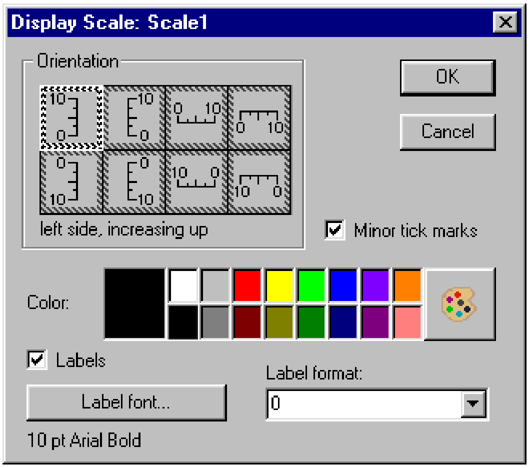
Minor unit = 2.5

Absolute Minimum and **Absolute Maximum** are numeric constants. They define the fullest possible range that the scale can show. These values act as clamps, restricting **Minimum** and **Maximum**.

Minimum and **Maximum** are numeric expressions you can use to change the minimum and maximum values on the scale. In the preceding example, the highest value of the scale (**Maximum**) is 400 if `Pot1` is less than 400, 1600 if `Pot1` is greater than 1600 (because of the **Absolute Maximum**), or equal to the value of Pot1.

Major unit specifies the number of units between major tick marks. **Minor unit** specifies the number of units between minor tick marks.

When you click on **OK**, the Display Scale dialog box appears.



Orientation: left side, increasing up

☑ Minor tick marks

☑ Labels

Label format: 0

10 pt Arial Bold

Specify **Orientation**, **Color**, **Label format**, and **Label font** as you choose.

You can remove minor tick marks by deselecting the **Minor tick marks** check box and you can remove label numbers from your scale altogether by deselecting the **Labels** check box. (Only major units have numeric labels.)

Scale Data Members

Table 2-49. Scale Data Members

Data Member	Type	Read	Write	Description
visible	logical	no	yes	When TRUE, the Scale becomes visible on the control panel. When FALSE, it is invisible. The default value is TRUE.

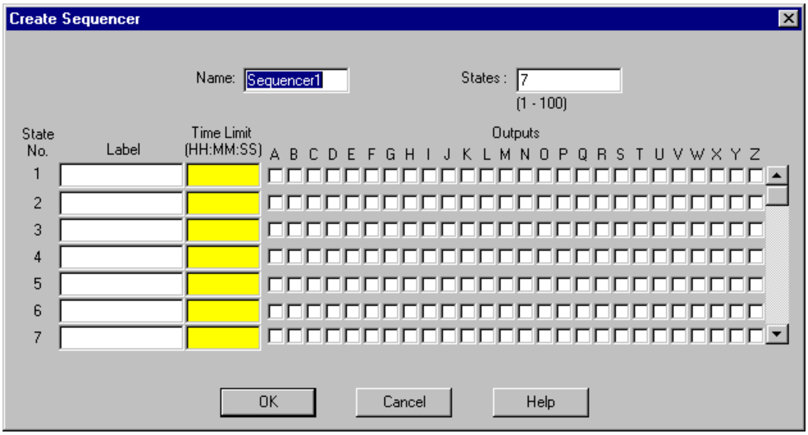
Comments Many people use this object class in conjunction with HyperTrends that are configured for a variable Y axis. They configure the **Minimum** and **Maximum** parameters of the Scale object to follow the same values as the **Max** and **Min** data members of the HyperTrend Object.

Sequencer

The Sequencer object is a method for cycling through a collection of up to 100 differently configured ON/OFF states for your process.

Each Sequencer state has 26 logical outputs. Outputs left blank are considered OFF.

You set each output to ON or OFF for each state. You can also create jumps to skip from one state to another, or activate immediate jumps to a designated state.



Name: Sequencer1

States : 7
(1 - 100)

State No.	Label	Time Limit (HH:MM:SS)	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1			□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
2			□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
3			□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
4			□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
5			□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
6			□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
7			□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□

The **States** field activates the states you want in your sequence loop. If you set **States** to `5`, the sequence object will pass through states 1 through 5 and then return to state 1 to repeat the cycle.

Notice that unused states can still be configured. In this way you can preconfigure special states, create subroutines, and so on. See the *Programming the Sequencer* section for further information on using this object.

Use **Label** to label each state.

The **Time Limit** parameter determines how long the Sequencer holds each state. If you leave this parameter blank, the Sequencer halts on that state until a `Goto` or `Jump` is activated, or the **Time Limit** parameter is changed.

Note When you set the Time Limit expression to reference a Lookout*Direct* control, you must make sure the output of that control is either in HH:MM:SS format, or in days.

Outputs are the ON/OFF settings for each state.

Programming the Sequencer

Left unmodified, the Sequencer cycles through the states selected in the **States** field of the **Sequencer parameter configuration** dialog box, beginning with state 1. If you enter `4` in the **States** field, the Sequencer will cycle through states 1—4 continuously, spending the amount of time in each state set by the **Time limit**.

Use the `Goto` data members to jump to a given state of the Sequencer immediately. Use the `Jump` data member to skip automatically from one state to a target state.

There is a tiny lag in state transition when you use the `Jump` data member. For instance, if you have a sequence cycling through states 1—8, and under some circumstances you want to skip states 5 and 6, moving directly from state 4 to state 7, set the `jump5.7` data member to `TRUE`. When the Sequencer reaches the end of the state 4 **Time limit**, it switches to state 5, and then immediately jumps to state 7. State 5 will be active for a brief period (about 10 ms), however, which you should take into account in designing your sequences.

You can use sequential jumps if you choose. When you activate a jump from state 1 to state 5, for example, and also a jump from state 5 to state 9, the Sequencer skips from state 1 to state 9 with a 10 ms delay at state 5.

If you use a `Jump` or `Goto` data member to activate a state outside the number you selected in your **States** field, the Sequencer will continue to cycle through any states following the target state, until it either reaches a state with no set **Time limit**, or state 100.

If the Sequencer reaches a state with no set **Time limit**, it remains in that state until a `Goto` or `Jump` is activated, or until the **Time Limit** parameter is changed.

If the Sequencer reaches state 100, and if state 100 has a **Time Limit** set, the Sequencer cycles to state 1 at the end of that time, resuming its cycle inside the states defined by the **States** parameter.

If you use a `Goto` or `Jump` data member to activate a state outside the ordinary cycle of a Sequencer object, you must use the `Jump` or `Goto` data members to return the sequence to its automatic cycle.

Sequencer Data Members

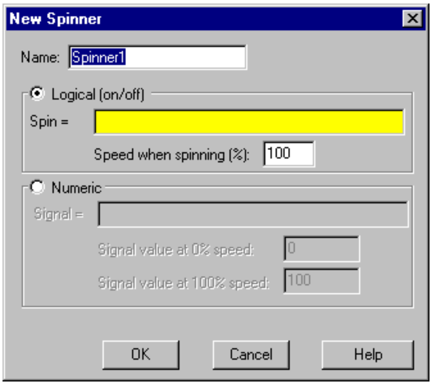
See the *Programming the Sequencer* section for detailed information on using Sequencer data members.

Table 2-50. Sequencer Data Members

Data Member	Type	Read	Write	Description
A – Z	logical	yes	no	Sequencer outputs
Goto1 – Goto100	logical	no	yes	When activated, forces Sequencer to go to the indicated state
Jump1.1 – Jump100.100	logical	no	yes	When activated, jumps from the state indicated by the first number to the state indicated by the second number
StateName	text	yes	no	Reports the currently active state name (**Label**)
StateNumber	text	yes	no	Reports the sequence number (**State No.**) of the current state
Time	numeric	yes	no	Reports the current length of time the Sequencer has been in the current state
TimeLimit	numeric	yes	no	Reports the length of time set for the Sequencer to hold the current state

Spinner

Spinner is a small, rotating disk. Its rotation speed can be variable, to represent the magnitude of a numeric **Signal**, or its rotation can be turned on or off based on the logical signal, **Spin**.



Name: Spinner1

● Logical (on/off)

Spin =

Speed when spinning (%): 100

○ Numeric

Signal =

Signal value at 0% speed: 0

Signal value at 100% speed: 100

Logical (on/off) and **Numeric** choose whether the spinner responds to a logical signal or a numeric signal. Choose **Logical** if you want to be able to turn the spinner on and off. Choose **Numeric** if you want the speed and direction of the spinner to change depending on a numeric variable.

Spin is a logical expression. When the result of the logical expression is TRUE, the spinner rotates at the rate defined by the **Speed when spinning (%)** field. **Speed when spinning (%)** is a numeric constant, ranging from –100 to 100.

Connecting the spinner to a positive value rotates the spinner in a counterclockwise direction. A negative value rotates the spinner in a clockwise direction.

Signal is a numeric expression. The result of this expression dictates the spin speed based on the linear range defined by **Signal value at 0% speed** and **Signal value at 100% speed**.

Table 2-51. Spinner Data Members

Data Members	Type	Read	Write	Description
none	—	—	—	Spinner does not have any data members

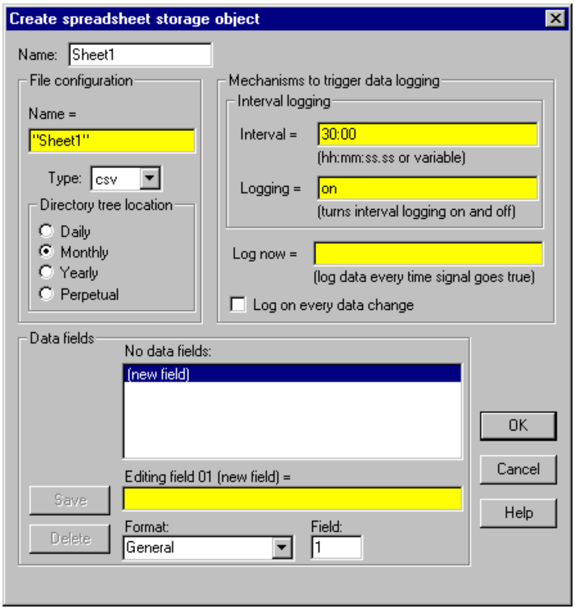
Comments Spinners are typically used to represent flow through a line or to show a motor running.

Spreadsheet

Spreadsheet permanently stores data to disk in spreadsheet files. You can log data on even and uneven intervals, when a data value changes, when an event occurs, or when any one of these things happen. Hence, you can implement complex logging criteria to meet almost any data storage need.

After each log, a new row is automatically added to the spreadsheet file. Lookout*Direct* can log a new row of data approximately 10 times per second; however, the time stamps associated with each row are rounded to the nearest second.

Each spreadsheet file may store any number of data signals. Each data signal is assigned a spreadsheet column, beginning with column number two. The first column contains the date and time. The first row contains the expressions associated with the data in each column. You may create any number of Spreadsheet objects for a given process.



Create spreadsheet storage object

Name: Sheet1

File configuration

Name = "Sheet1"

Type: csv

Directory tree location

○ Daily
● Monthly
○ Yearly
○ Perpetual

Mechanisms to trigger data logging

Interval logging

Interval = 30:00 (hh:mm:ss.ss or variable)

Logging = on (turns interval logging on and off)

Log now = (log data every time signal goes true)

□ Log on every data change

Data fields

No data fields:

(new field)

Editing field 01 (new field) =

Format: General

Field: 1

Name is the filename used to create a spreadsheet file. Lookout*Direct* assigns a DOS filename to each spreadsheet file by adding the **Type** extension to the **Name**. Currently, Lookout*Direct* supports only one **Type**: comma separated value format (`.csv`). Most database and spreadsheet programs including Microsoft Excel directly read the `.csv` file format.

Because the **Name** parameter is a text expression field, you can create new `.csv` files with unique names dynamically. This is especially useful for recording batch processing data. The definition dialog box Figure is configured so that an operator can enter a batch name using a TextEntry object before the batch is started. The text expression appends the filename to the specified path, `C:\BATLOG\`. So if the operator enters a file name like `BATCH71`, then the full path name would be `C:\BATLOG\BATCH71.CSV`. When the `BatchRun` logical signal goes TRUE, Lookout*Direct* creates the new `.csv` file and begins writing to it. When `BatchRun` goes FALSE, logging ends, leaving a comprehensive log of all data associated with the batch.

This example forces Lookout*Direct* to store the `.csv` file in a particular directory because it specifies a full path name. If you enter a relative pathname like `"\BATLOG\"&TextEntry1`, the file is located in that subdirectory of the identified *Direct***ory tree location**. So, for example, the full path name of the file might be `C:\LOOKOUT\1995\SEP\BATLOG\BATCH71.CSV`.

If you enter just a filename such as `"DATA"`, the file location is based on the Lookout*Direct* directory. For example, the full path name of the file might be `C:\LOOKOUT\1995\SEP\DATA.CSV`.

If you select **Daily**, Lookout*Direct* creates a new file and subdirectory every day in which to store the data. If you select **Monthly**, Lookout*Direct* creates a new file and subdirectory every month in which to store the data. If you select **Yearly**, Lookout*Direct* creates a new file and subdirectory every year. **Perpetual** files are stored in the root directory as specified by your **Data files location** parameter.

The following examples are the DOS filenames and directory trees created by Lookout*Direct* for a spreadsheet file named, `DATA`.

Daily	Yearly
`c:\lookout\1993\sep\09\data.csv`	`c:\lookout\1993\data.csv`
`\sep\10\data.csv`	`\1994\data.csv`
`\sep\11\data.csv`	`\1995\data.csv`
Monthly	**Perpetual**
`c:\lookout\1993\sep\data.csv`	`c:\lookout\data.csv`
`\oct\data.csv`	
`\nov\data.csv`	

The **Mechanisms to trigger data logging** are a set of tools used to create a simple or complex logging scheme, as desired. Use these parameters to log data based on a timer, event, or any combination of the two. When the spreadsheet is triggered, all data in the Data fields is logged to disk.

Interval is a numeric expression used to create a Pulse timer with a pulse period of the specified time period and a pulse duration of zero. Normally this is a time formatted constant value such as 15:00 (fifteen minutes), for example.

Logging is a logical expression that turns the **Interval** parameter on and off. It could be a switch on a control panel, a logical input from an external device, or a more complex expression. Normally this is a constant value, `ON` or `OFF`.

Log now is a logical expression. When **Log now** transitions from OFF to ON, Lookout*Direct* logs the data. A transition from ON to OFF has no effect. This expression could be a pushbutton on a control panel, a logical signal from a device, or a more complex expression.

The **Log on every data change** option should be used with care. When turned on, it triggers the logging of data any time any one of the data fields experiences a change. This is normally used to log the starting and stopping of pumps, opening and closing of valves, or other similar events. If your data fields contain even a single analog value that changes often, you could end up triggering the logger thousands of times. Or if they contain a logical value that changes frequently, you could have the same problem.

The **Data fields** window lists all expressions that have been entered for logging in the order of their field number.

The **Save** button saves your new or modified expression in the **Data fields** window along with a new field number if any. Normally, **Data fields** contain simple expressions like `PLC1.Tanklevel`.

The **Delete** button deletes the currently selected expression from the data fields list.

The **Format** option specifies the numeric format assigned to the currently selected numeric expression when it is logged to disk. This has no effect on logical or text expressions.

Field indicates the number of the currently selected data field.

Note Field numbers should not be modified after data has been stored or the data will not appear under correct headers until a new file is created.

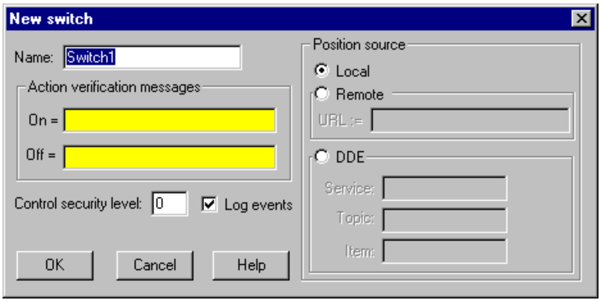
Spreadsheet Data Members

Table 2-52. Spreadsheet Data Members

Data Members	Type	Read	Write	Description
logged	logical	yes	no	Spreadsheet file update pulse. The Spreadsheet object generates this logical pulse with a pulse duration of zero after each successful log.

Switch

Switch generates a logical signal for receipt by other objects. Switches change state when you click on them with a mouse button, trackball, touchscreen, or space bar on your keyboard.



New switch

Name: Switch1

Action verification messages

On =

Off =

Control security level: 0

☑ Log events

Position source

● Local

○ Remote

URL :=

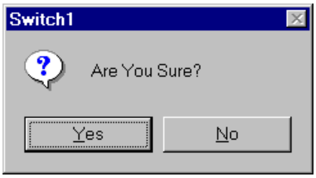
○ DDE

Service:

Topic:

Item:

Use **Action verification messages** to create dynamic text expressions to be displayed in message dialog boxes. See Chapter 6, *Security*, in the Lookout*Direct Developer's Manual* for more information on security.



Switch1

Are You Sure?

Position source determines where the value of the Switch resides. **Local** indicates the value of the Switch lies within the object itself—on the control panel. If the switch is up the signal is ON, if down the signal is OFF.

Remote Switches get their values from a remote source, often the register on a controller they are connected to. Flipping the Switch changes the status of the register, and changing the status of the register flips the switch.

The **Remote** option is especially useful when you want to prevent Lookout*Direct* from changing the value of setpoints or registers upon initial startup, or reconnection of lost communication.

The **Remote** option calls for a URL to locate the data member you want to connect to. The URL field is green, and you cannot use a complex expression as a URL. If you need to use one RadioButton for several purposes, you can use a Symbolic Link to make a more complex connection than that possible with a URL.

You can right-click in the **URL** field and use the **URL Editor** dialog box to assemble the URL, in the same way you use the Lookout*Direct* expression editor.

A remote position source connection is completely reciprocal. A change in your Lookout*Direct* control changes the data member that control is remoted to. Any change in that data member also changes the control. It is not necessary, and is incorrect, to use the **Edit Connections** dialog box to connect a control object to its controlled data member.

Because the remote connection is reciprocal, you can only make such a connection to a data member that is either writable, or readable and writable.

When a process with remoted controls first opens, those controls take their initial values by following the URL to read the data members they are remoted to. The control will be covered by a red X to indicate that the remote connection is not functioning.

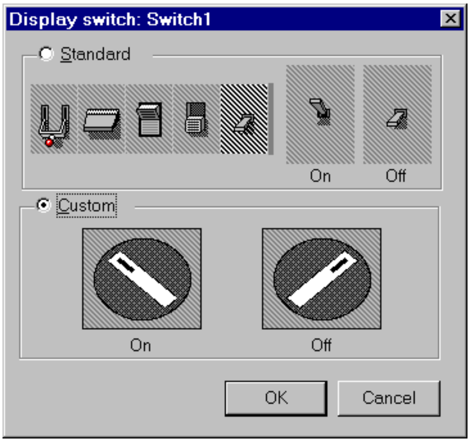
Note You should use **Remote** to connect a control in a client process to a data member in a server process. Connections inside a single process can be made using **Object»Edit Connections**.

Because complex expressions are read-only values, you cannot remote directly to them. For the same reason, you cannot remote one control to another control's `(intrinsic)` data member (though you can remote a control to another control's value data member).

Much like Remote Switches, **DDE** (Dynamic Data Exchange) Switches get their values from a remote source. This could be a cell in a spreadsheet, another DDE aware application, or a second copy of Lookout*Direct* running on the network. See Chapter 5, *Dynamic Data Exchange*, in the Lookout*Direct Developer's Manual* for more information on **Service**, **Topic**, and **Item** parameters.

Control security level specifies the minimum security level operators must have to gain access to this individual object, and thus control it.

The **Log events** option creates a permanent audit trail for the object—who did what and when. All adjustments of the Switch are logged to disk, including the time the Switch was flipped, the operator's account name, and the direction the Switch was flipped. See Chapter 7, *Logging Data and Events*, in the Lookout*Direct Developer's Manual* for more information on event logging.



You can replace the standard switch types with custom graphic symbols. If you decide to use custom graphics, you must specify both symbol parameters, **On** and **Off**. See Chapter 2, *Graphics*, in the Lookout*Direct Developer's Manual* for more information on creating custom graphic symbols and the use of transparent pixels.

Switch Data Members

Table 2-53. Switch Data Members

Data Members	Type	Read	Write	Description
(implicit)	logical	yes	no	Switch Position
enable	logical	no	yes	If TRUE (the default), enables DDE. If FALSE, disables DDE. The default value is ON. This input is ignored for non-DDE TextEntry objects.
reset	logical	no	yes	While this value equals TRUE, the control will be set to the value in `resetvalue`.
resetvalue	numeric	no	yes	Sets the value a control will take when the reset data member transitions from FALSE to TRUE.

Table 2-53. Switch Data Members (Continued)

Data Members	Type	Read	Write	Description
value	numeric	yes	yes	The current value of the control. If you have remoted this control, then `value` is the current value of the position source.
visible	logical	no	yes	When FALSE, the switch object cannot be seen on the display panel. When TRUE, the Switch can be seen and controlled.

Comments If a switch with more than two positions is needed, use a Pot object instead.

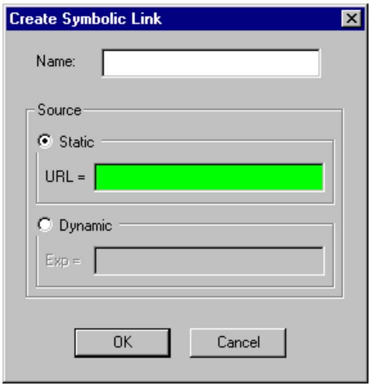
Related Objects *Pushbutton, Pot*

Symbolic Links

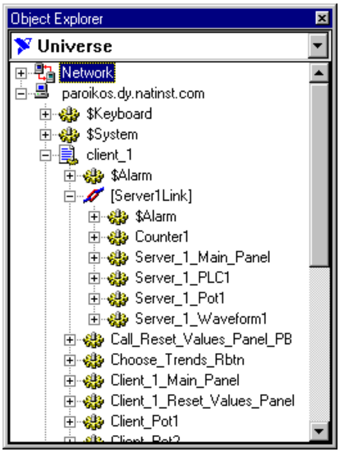
A Symbolic Link is a Lookout*Direct* object you use to make certain kinds of remote connections easier and more efficient. This object can also be used to manage failover redundancy in Lookout*Direct* 4. (Redundancy is not implemented in the Lookout*Direct* 4 Preview.)

The Symbolic Link serves as a flexible intermediary between separate processes whether they are running on one computer or on different computers.

Create a Symbolic Link using the Lookout*Direct* object browser. Right-click on the process or folder into which you want to insert the new Symbolic Link. The following dialog box appears.



The Symbolic Link can represent either a static or a dynamic source. The **Static** source is a URL pointing to a computer, process, folder, or object running in some instance of Lookout*Direct* on your network. A typical Symbolic Link set to a process appears in the following illustration.



As you can see, all the objects in the `Server_1` process are represented in the Symbolic Link `[Link_to_Server1]`. You can drag and drop any of the objects or data members onto a panel in the process containing the Symbolic Link, where they will appear as expressions.

The **Dynamic** source must always be an expression that evaluates as a text string. When you select this option, you can use text strings and text variables to prepare the Symbolic Link to be used by a control for remote connections that cannot be made directly.

For example, you cannot use a complex expression in a Pot control remote source URL. You can, however, construct a complex expression as the dynamic source in a Symbolic Link, and then set the URL to connect to that Symbolic Link.

For instance, the following expression evaluates as two different URLs depending on the position of `Switch1` (line breaks inserted for clarity).

```
tif(Switch1,
"\\.\server_1\Server_1_Waveform1.sinewave",
"\\.\Server2\Server2_Waveform.sinewave")
```

If you connect a HyperTrend item to the Symbolic Link containing this dynamic source, changing the switch changes which server provides the wave form being plotted on the HyperTrend graph.

As another example, because Lookout*Direct* data is now polymorphic, you can construct strings using both text and numeric inputs.

For example, if you create Pot1 with a **minimum** of `0`, a **maximum** of `9`, and an **interval** of `1`, you can use the expression

```
"Modbus1.4001" & Pot1
```

as the entry for the **Dynamic** option in a Symbolic Link.

You can then use the Symbolic Link to connect a control to Modbus data members `40010-40019`, depending on the setting of Pot1.

While you could not directly remote your control to that range of Modbus data members, you can make a remote connection from that control to the Symbolic Link using that dynamic expression.

If you connect the **Dynamic** expression to a data table, you have a nearly unlimited ability to have one object control a large number of data members, either through a cursor control or a radio button control.

$System

$System is a global object. It makes global Lookout*Direct* data such as the currently logged in user name and security level available for use in your process. You can use $System data members just like other object data members.

The **seclevel** data member is always an integer value between 1 and 10. This number represents the Lookout*Direct* security level of the user currently logged in. For more information about Lookout*Direct* security levels, see Chapter 6, *Security*, in the Lookout*Direct Developer's Manual*.

The **time** data member represents the current date and time of the system. Like all time values in Lookout*Direct*, this is a floating point number in which the integer represents the date and the fraction represents the time of day. You can use the various Lookout*Direct* date and time numeric formats to view this value in the most convenient format. This data member updates itself every minute, on the minute. It also updates itself immediately after it is created or when its process is opened.

The **username** data member is the account name of the user currently logged in. For more information about Lookout*Direct* security accounts, see Chapter 6, *Security*, in the Lookout*Direct Developer's Manual*.

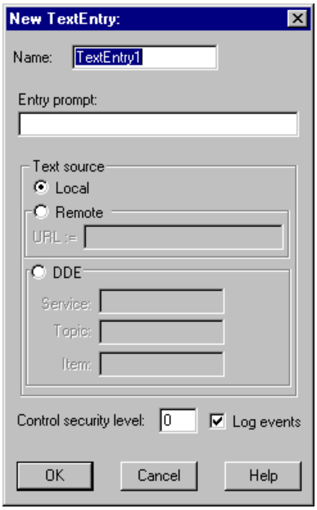
$System Data Members

Table 2-54. $System Data Members

Data Member	Type	Read	Write	Description
NetworkStatus	text	yes	no	Reports the status of the last network transaction of your computer. You may need to use this data member in conjunction with a Sample object to keep track of high levels of network activity with many I/O points.
seclevel	numeric	yes	no	Security level of the user currently logged in
time	numeric	yes	no	Current operating system time
username	text	yes	no	Name of the user currently logged in

TextEntry

With TextEntry you can manually enter textual notes with the keyboard. These notes may contain any combination of numeric and alphanumeric characters; however, the result of your entry is converted to a text value. Just like any other text expression in Lookout*Direct*, your note can be logged to disk, connected to other data members that accept text signals, and so on. The note is saved and displayed as a single line entry—you cannot embed carriage returns into the message.



New TextEntry:

Name: TextEntry1

Entry prompt:

Text source

● Local

○ Remote

URL :=

○ DDE

Service:

Topic:

Item:

Control security level: 0 ☑ Log events

Entry prompt is the text that appears at the top of the text entry dialog box when an operator selects the text entry pushbutton.

Text source determines where the user-entered text resides. **Local** indicates the user-entered text lies within the object itself—on the control panel.

Remote indicates that the user-entered text resides in a remote source, such as a text expression or another TextEntry object.

The **Remote** option calls for a URL to locate the data member you want to connect to. The URL field is green, and you cannot use a complex expression as a URL. If you need to use one RadioButton for several purposes, you can use a Symbolic Link to make a more complex connection than that possible with a URL.

You can right-click in the **URL** field and use the **URL Editor** dialog box to assemble the URL, in the same way you use the Lookout*Direct* expression editor.

A remote position source connection is completely reciprocal. A change in your Lookout*Direct* control changes the data member that control is remoted to. Any change in that data member also changes the control. It is not necessary, and is incorrect, to use the **Edit Connections** dialog box to connect a control object to its controlled data member.

Because the remote connection is reciprocal, you can only make such a connection to a data member that is either writable, or readable and writable.

When a process with remoted controls first opens, those controls take their initial values by following the URL to read the data members they are remoted to. The control will be covered by a red X to indicate that the remote connection is not functioning.

Note You should use **Remote** to connect a control in a client process to a data member in a server process. Connections inside a single process can be made using **Object»Edit Connections**.

Because complex expressions are read-only values, you cannot remote directly to them. For the same reason, you cannot remote one control to another control's `(intrinsic)` data member (though you can remote a control to another control's value data member).

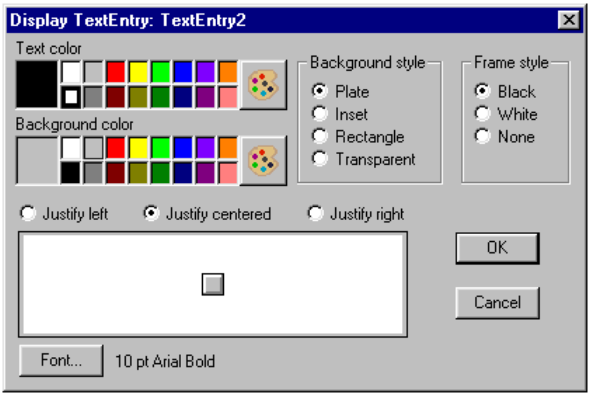
Much like **Remote** TextEntry objects, **DDE** TextEntry objects get their values from a remote source. This is the option you use to tie the text to a cell in a spreadsheet, a database lookup table, or any DDE aware application—including a second copy of Lookout*Direct* running on the network. See Chapter 5, *Dynamic Data Exchange*, in the Lookout*Direct Developer's Manual* for more detailed information on **Service**, **Topic** and **Item**.

Note The last DDE parameters used on any object automatically become the default values for any new DDE object.

Control security level specifies the minimum security level operators must have to gain access to this individual object, and thus control it.

The **Log events** option creates a permanent audit trail for the object—who did what and when. When selected, all text entries in this object are logged to disk. Each entry includes the time of the entry, the operator's account name, and what entry was made. See Chapter 7, *Logging Data and Events*, in the Lookout*Direct Developer's Manual* for more information on event logging.

Lookout*Direct* presents the following display parameters dialog box after you define the object. It lets you define the text font and presentation style.



Display TextEntry: TextEntry2

Background style: ● Plate ○ Inset ○ Rectangle ○ Transparent

Frame style: ● Black ○ White ○ None

○ Justify left ● Justify centered ○ Justify right

Font... 10 pt Arial Bold

TextEntry Data Members

Table 2-55. TextEntry Data Members

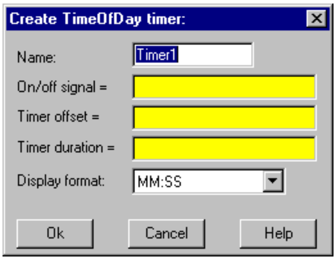
Data Members	Type	Read	Write	Description
(implicit)	text	yes	no	Current, user-entered text
enable	logical	no	yes	If TRUE (the default), enables DDE. If FALSE, disables DDE. The default value is ON. This input is ignored for non-DDE TextEntry objects.
reset	logical	no	yes	While this value equals TRUE, the control will be set to the value in `resetvalue`.

Table 2-55. TextEntry Data Members

resetvalue	numeric	no	yes	Sets the value a control will take when the reset data member transitions from FALSE to TRUE.
value	numeric	yes	yes	The current value of the control. If you have remoted this control, then `value` is the current value of the position source.

TimeOf*xxxx*

TimeOf*xxxx* are timers that generate a periodic pulse of a specified duration. The timers are turned on and off by **On/off signal**. The time period is defined by the type of timer used—a TimeOfMinute timer has a one-minute period, a TimeOfYear timer has a one-year time period, and so on. The output of these timers goes high after the specified **Timer offset** has elapsed in the current period and remains high for the specified **Timer duration**.



Create TimeOfDay timer:

Name: Timer1

On/off signal =

Timer offset =

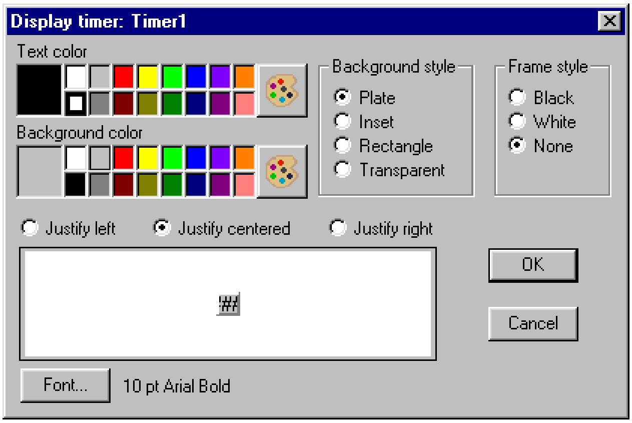
Timer duration =

Display format: MM:SS

The **Timer offset** and **Timer duration** can range from 0.0 seconds to a year, and the effective resolution is 0.01 seconds over the entire range. The **Timer offset** plus the **Timer duration** should always be less than or equal to the time period.

The object display shows the time remaining before the output changes state and is updated approximately once per second. It is shown in the selected **Display format**. If the **On/off signal** is OFF, the display shows `OFF`.

The **On/off signal** is a logical expression while **Timer offset** and **Timer duration** are numeric expressions. Normally, these are simple time constants such as 6:10:20 (six hours:ten minutes:twenty seconds). See *Numeric Data Members* in Chapter 2, *How* Lookout*Direct Works*, in the *Getting Started with* Lookout*Direct* manual for more information on entering time constants.



Display timer: Timer1

Background style
● Plate
○ Inset
○ Rectangle
○ Transparent

Frame style
○ Black
○ White
● None

○ Justify left ● Justify centered ○ Justify right

10 pt Arial Bold

Timeof*xxxx* Data Members

Table 2-56. TimeOf*xxxx* Data Members

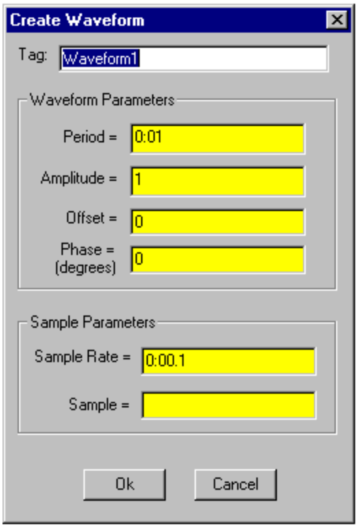
Data Members	Type	Read	Write	Description
(implicit)	logical	yes	no	Logical timer value

Comments TimeOf*xxxx* can be used in place of Pulse objects when the pulse needs to be synchronized with the clock—if a pump should only be allowed to run between the hours of 8:00 and 17:00 each day, the TimeOfDay timer should be used.

Related Objects *DelayOff, DelayOn, Interval, OneShot, Pot*

Waveform

The Waveform object generates cosine, sine, square, sawtooth, triangle, and random waveforms.



Create Waveform

Tag: Waveform1

Waveform Parameters

Period = 0:01

Amplitude = 1

Offset = 0

Phase = (degrees) 0

Sample Parameters

Sample Rate = 0:00.1

Sample =

The Waveform object produces output by sampling the desired waveform at the specified **Sample Rate**. The **Period** (in days), **Amplitude**, **Offset** (level shift), and **Phase** (in degrees) of the generated waveforms can be set to any numerical expression. A single Waveform object can be used to generate multiple waveforms of varying relative phases, all with the same **Period**, **Amplitude**, **Offset** and absolute **Phase**.

Lookout*Direct* samples a waveform when the logical expression **Sample** transitions from FALSE to TRUE. This can be a simple expression like the signal from a pushbutton, or it can be a complex algorithm.

Sample Rate is a numeric expression that determines how often to sample the waveform. Lookout*Direct* converts the numeric value of **Sample Rate** into a time signal that represents days and fractions of a day. The Waveform object then samples the waveforms at the specified time interval. Normally, this will be a simple time constant such as 0:01 (one second).

When the **Period** of the waveform is changed, the waveform must be phase shifted so that there is a smooth transition to the new frequency. This is handled internally by the Waveform object and does not effect the **Phase** parameter.

As a result, multiple waveform objects will not be in phase with each other if one or more have had their **Periods** changed. To reset a waveform so that it will again be in phase with other waveforms, use the **Reset** connection. See *Waveform Comments* for a more detailed discussion of phase.

Each waveform output (except random) has 361 separate data members for generating waveforms of different relative phase shifts. For example, cosine0 (or just cosine) is a cosine wave of 0-degree relative phase shift, cosine90 is a cosine wave with +90 degrees relative phase shift, and cosine_90 is a cosine wave with –90 degrees relative phase shift.

Waveform Data Members

Table 2-57. Waveform Data Members

Data Members	Type	Read	Write	Description
cosine_180 — cosine180	numeric	yes	no	Cosine waveform output.
random	numeric	yes	no	Random waveform output.
Reset	logical	no	yes	Resets the phase shift to zero when transitioned from OFF to ON. (See comments on phase in Waveform Comments.)
Sample	logical	no	yes	When this transitions from false to true, the waveform is sampled.
SampleRate	numeric	no	yes	Specifies the rate at which the waveform will be automatically sampled.
saw_180 — saw180	numeric	yes	no	Sawtooth waveform output.
sine_180 — sine180	numeric	yes	no	Sine waveform output.
square_180 — square180	numeric	yes	no	Square waveform output.
triangle_180 — triangle180	numeric	yes	no	Triangle waveform output.

Waveform Comments

A waveform is a function of time whose values repeat every period. The total phase shift of the waveform determines where, in its cycle, the waveform starts at time t = 0 (midnight on the morning of the January 1, 1900 for Lookout*Direct*). Two waveforms that start at the same place in their cycle at t = 0 (that is, same total phase shift) are said to be “in phase” with each other.

For a waveform generated by a Waveform object, the total phase shift is equal to the absolute phase shift as specified by the **Phase** parameter, and by the relative phase shift specified by the selection of a particular data member (for example, cosine 90). Because the phase is defined in terms of absolute time, two waveform objects with the same **Period** and **Phase** will generate waveforms that are in phase with each other.

However, if the **Period** of one of the waveforms is variable (for example, if it is connected to a Pot object) and changes, the absolute phase of the waveform will be changed. This additional phase shift is handled internally by the Waveform object, and is only noticeable by the fact that it can produce an undesirable phase shift between two Waveform objects that have the same **Period**. For this reason, you may want to use the **Reset** data member to reset this internal phase shift.

Note The **Phase** parameter is in degrees. A **Phase** of 180 degrees will shift the waveforms one half period ahead. For example if the **Period** is 1 day and the **Phase** is 180 degrees, then the shift will be one-half day.

$$\frac{\textbf{Phase}}{360^\circ} \times \textbf{Period} = \frac{180^\circ}{360^\circ} \times 1 day = \frac{1}{2} day$$

3

Driver and Protocol Objects

This chapter describes Lookout*Direct* Driver and protocol object classes, listed in alphabetical order. Input parameter syntax and data members are documented for each object class, along with a description of the functionality of each object class.

AB_PLC5, AB_SLC500

Lookout*Direct* uses the AB object classes to communicate with the Allen-Bradley family of PLC controllers using a variety of interfaces.

Lookout*Direct* can communicate with a member of the PLC-2 family in the following ways:

- Through a Data Highway Plus (DH+) connection to an Allen-Bradley 1785-KA3 PLC-2 adapter module using an Allen-Bradley 1784-KT, 1784-KTx, or 1784-PCMK card, or an S-S Technologies 5136-SD direct-link interface card installed in the computer,
- Through the serial port using an Allen-Bradley KF2 module (which converts serial DF1 to DH+) to an Allen-Bradley 1785-KA3 PLC-2 adapter module, or
- Through a direct DF1 serial connection to the PLC programming port.

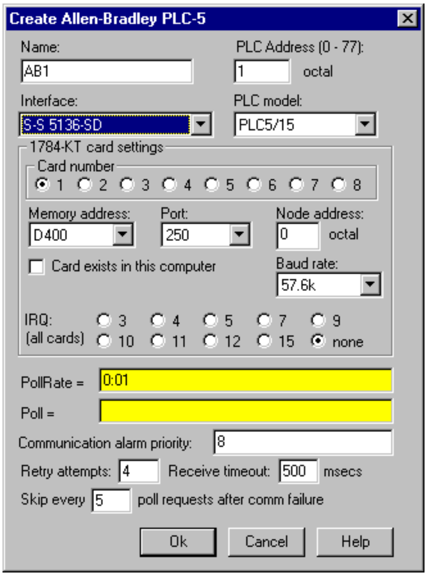
Lookout*Direct* can communicate with a member of the PLC-5 family in the following ways:

- Through a direct Ethernet connection to the PLC AUI port,
- Through a direct DH+ connection using a 1784-KT, 1784-KTx, 1784-PCMK, or 5136-SD card installed in the computer,
- Through the serial port via a KF2 module which converts serial DF1 to DH+, or
- Through a direct DF1 serial connection to the PLC programming port.

Lookout*Direct* can communicate with a member of the SLC-500 family in the following ways:

- Through a direct DH+ connection using a 1784-KT, 1784-KTx, 1784-PCMK, or 5136-SD card installed in the computer,
- Through the serial port via a KF2 module which converts serial DF1 to DH+,
- Through the serial port using an Allen-Bradley 1747-KE card (which plugs into the SLC chassis and converts DF1 to DH 485),
- Through the serial port using a stand-alone Allen-Bradley 1770-KF3 communication interface module which converts DF1 to DH 485,
- Through a direct DF1 serial connection to a SLC 5/03 or SLC 5/04 programming port, or

- Through a direct DH485 connection using a 1784-PCMK card in conjunction with either a 1747-AIC or 1761-NET-AIC module.



Create Allen-Bradley PLC-5

Name: AB1

PLC Address (0 - 77): 1 octal

Interface: S-S 5136-SD

PLC model: PLC5/15

1784-KT card settings

Card number ● 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8

Memory address: D400

Port: 250

Node address: 0 octal

□ Card exists in this computer

Baud rate: 57.6k

IRQ: (all cards) ○ 3 ○ 4 ○ 5 ○ 7 ○ 9 ○ 10 ○ 11 ○ 12 ○ 15 ● none

PollRate = 0:01

Poll =

Communication alarm priority: 8

Retry attempts: 4 Receive timeout: 500 msecs

Skip every 5 poll requests after comm failure

PLC Address refers to the PLC network node address setting as configured on the physical device. If devices share a common **Interface**, they require unique addresses. When using DF1 protocol (serial communications), valid addresses range from 0 to 254 decimal. When using DH+, valid addresses range from 0 to 77 octal.

PLC Model specifies the particular type of PLC or SLC you are representing with this object. The **PLC Model** you select determines what native data members comprise the object.

PollRate is a numeric expression that determines how often to poll the device. Lookout*Direct* polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *How LookoutDirect Works*, of the *Getting Started with LookoutDirect* manual for information on entering time constants.

Poll is a logical expression. When this expression changes from false to true, Lookout*Direct* polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of the alarms generated by the AB object.

Retry attempts specifies the consecutive number of times Lookout*Direct* attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the object generates an alarm and begins to **Skip every *n* poll requests after comm failure**. Once Lookout*Direct* reestablishes communications, it polls the device on its regular cycle, as defined by **PollRate**.

Receive timeout is the time delay Lookout*Direct* waits for a response from a device before retrying the poll request.

Interface identifies the type of communication hardware you are using. The selection you make here determines what protocol parameters you have to specify. The paragraphs that follow describe interface-specific protocol parameters.

Allen-Bradley Serial Port Interface Parameters

The `KE/KF/Serial` **Interface** selection enables serial port communication via a KE card, a KF3 module or KF2 module. When using your serial port, Lookout*Direct* employs the Allen-Bradley full-duplex (peer-to-peer) DF1 protocol. Figure shows an Allen Bradley object configured for serial communications.

Serial port specifies which RS-232C port the object uses for communication to the physical device.

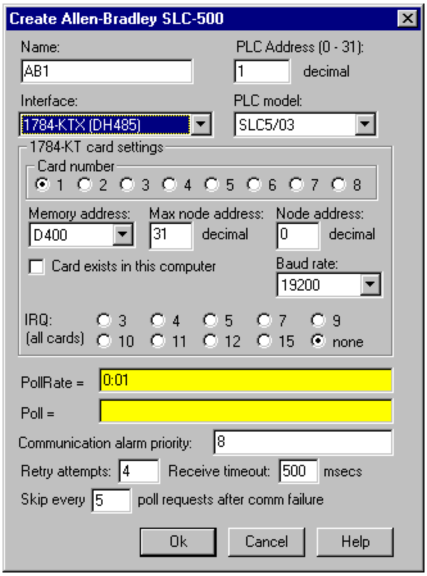
Data rate, **Parity**, and **Error detection** reference the settings on the hardware device. The AB object classes support both *BCC* (block check character) and *CRC* (cyclic redundancy check) error detection. BCC provides a medium level of data security. CRC ensures a higher level of data security. Choose the settings as configured on your PLC or SLC.

Phone number specifies the number to be dialed if the serial port setting is configured for dial-up. This number only applies to the individual object.

Allen-Bradley DH+ Interface Parameters

Note When you configure an AB_PLC2, AB_PLC5, or AB_SLC500 for DH+, Lookout*Direct* creates a file called `ALLBRAD.INI`. This file contains the configuration settings that you enter for your KT card(s). If you plan to run the process file on a Lookout*Direct* Runtime System, be sure to copy the INI file along with your process (`.lkp`) file to the target computer.

The `1784-KT`, `1784-KTx`, `1784-PCMK`, and `S-S 5136-SD` **Interface** selections enable direct connection of your computer to a DH+ network. The following illustration shows an AB_SLC500 configured for DH+ communications using a 1784-KTx card.


Create Allen-Bradley SLC-500

Name: AB1

PLC Address (0 - 31): 1 decimal

Interface: 1784-KTX (DH485)

PLC model: SLC5/03

1784-KT card settings

Card number ● 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8

Memory address: D400

Max node address: 31 decimal

Node address: 0 decimal

□ Card exists in this computer

Baud rate: 19200

IRQ: (all cards) ○ 3 ○ 4 ○ 5 ○ 7 ○ 9 ○ 10 ○ 11 ○ 12 ○ 15 ● none

PollRate = 0:01

Poll =

Communication alarm priority: 8

Retry attempts: 4 Receive timeout: 500 msecs

Skip every 5 poll requests after comm failure

Card number selects which network interface card that the PLC is connected to in case your computer has multiple KT or S-S cards.

Card DH+ node address identifies the address of the interface card in the DH+ network. Valid addresses range from 0 to 77 octal. The node address of the card must be unique—that is, it must not be the same as the address of any other device on the DH+ network.

Memory address specifies the base address location of the selected interface card memory. Your selection should match the settings on the card. If you are using multiple interface cards, be sure each card has a unique address. The network interface cards use dual-ported memory.

For this reason, if you are using a memory manager such as EMM386 it is important to add a memory exclusion statement to your `CONFIG.SYS` file. The table on the following page lists base memory address selections and corresponding exclusions for all legal memory addresses for the 1784-KT card.

Use **Max node address** to maximize the performance of a DH485 network by assigning addresses to nodes on the network using consecutive numbers starting with zero. Set the **Max node address** to the maximum of the assigned node addresses. By default, the value of this variable is 31, which is the largest legal address for any node on a DH485 network.

The 1784-KT interface card has on-board network termination resistors. If you are using such a card and if your computer is the last node on the network and if the cable does not already have a terminating resistor on it, then select the **Enable link termination resistor** check box.

Use the **Card exists in this computer** check box to instruct Lookout*Direct* whether or not to look for the interface card in the computer. *Be sure to check this box when you are ready to start polling your PLCs.* When you check this box and select **OK**, Lookout*Direct* initializes the card, activates its self-test, and downloads its driver firmware. Then polling begins. Leave the **Card exists in this computer** check box deselected (this is the default setting) if the card is not in your computer (for example, if you are developing a process on a computer different from the one that will be running the process) or if you do not want to poll any PLC connected to the card.

If you deselect the **Card exists in this computer** check box, you are disabling communications using this interface card with all PLCs connected to it.

Table 3-1. Allen-Bradley DH+ Interface Memory Addresses

Memory Address	AB 1784-KT Exclude	AB 1784-KTx 1784-PCMK Exclude	SS-5136-SD Exclude*	Recommendation
A000	A300-A3FF	A000-A0FF	A000-A3FF	Typically used by VGA drivers. Use if no other option is available.
A400	A700-A7FF	A400-A4FF	A400-A7FF	
A800	AB00-ABFF	A800-A8FF	A800-ABFF	
AC00	AF00-AFFF	AC00-ACFF	AC00-AFFF	
B000	B300-B3FF	B000-B0FF	B000-B3FF	Used by MDA & CGA drivers. Use if no Dxxx option is available.
B400	B700-B7FF	B400-B4FF	B400-B7FF	
B800	BB00-BBFF	B800-B8FF	B800-BBFF	
BC00	BF00-BFFF	BC00-BCFF	BC00-BFFF	
C000	C300-C3FF	C000-C0FF	C000-C3FF	Typically used by BIOS. Use if no Dxxx option is available.
C400	C700-C7FF	C400-C4FF	C400-C7FF	
C800	CB00-CBFF	C800-C8FF	C800-CBFF	
CC00	CF00-CFFF	CC00-CCFF	CC00-CFFF	
D000	D300-D3FF	D000-D0FF	D000-D3FF	Normally available.Try to use one of these first.
D400	D700-D7FF	D400-D4FF	D400-D7FF	
D800	DB00-DBFF	D800-D8FF	D800-DBFF	
DC00	DF00-DFFF	DC00-DCFF	DC00-DFFF	

* The 5136-SD memory exclusions recommended here are based on 16K memory mapping. Because you are using the SS card to emulate the KT card, there is no advantage to using its 32K memory access capability.

Baud rate (1784-KTx, 1784-PCMK, 5136-SD only) selects the baud rate of the DH+ network. The default is 57.6k baud. Before selecting a higher baud rate, be aware of that only a few PLCs (such as the SLC5/04) support higher baud rates; that every node on a DH+ network must support the baud rate used on that network; that the maximum network cable length is smaller for higher baud rates; and that the correct values for the termination resistors at the ends of the network cable are different for higher baud rates. Consult the manuals that came with your hardware for more detailed information.

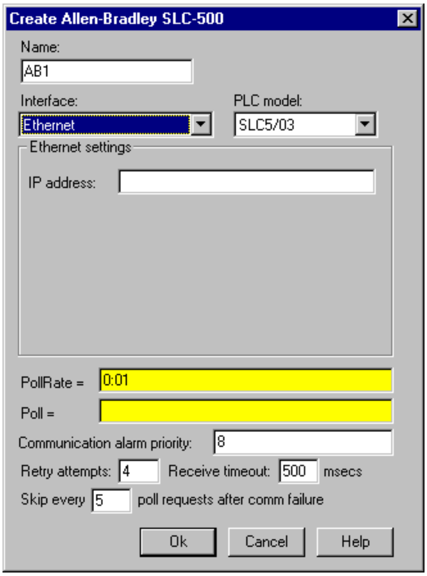
IRQ (all cards) identifies the interrupt setting of all DH+ interface cards installed in the computer. This selection should match the IRQ settings on *all* of the interface cards.

Assigning an interrupt to the interface card(s) improves overall computer performance somewhat. Any time one of the cards receives an input, it generates an interrupt recognized by Lookout*Direct*.

⚠ **Caution** Be sure to verify that no other drivers or cards are mapped to the selected memory address or use the same interrupt.

Allen-Bradley Ethernet Interface Parameters

The `Ethernet` **Interface** selection enables direct communication between your computer and a PLC using a standard Ethernet network. The following diagram shows an AB_PLC5 configured for Ethernet communications.



Create Allen-Bradley SLC-500

Name: AB1

Interface: Ethernet

PLC model: SLC5/03

Ethernet settings

IP address:

PollRate = 0:01

Poll =

Communication alarm priority: 8

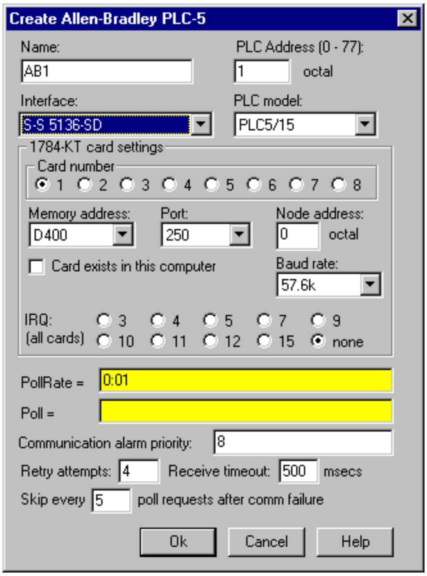
Retry attempts: 4 Receive timeout: 500 msecs

Skip every 5 poll requests after comm failure

IP address specifies the Internet protocol address of the PLC. An Internet protocol address consists of four numbers, separated by periods. Each number ranges from zero to 255 decimal. Thus, a typical Internet address might be 128.7.9.231. Ensure that the **IP address** you enter matches the Internet protocol address of the PLC your object represents. You can also enter the IP address by name.

Using the 5136-SD card from S-S Technologies, Inc.

To use the 5136-SD card, select the S-S 5136-SD interface in the **Create Object** dialog box. It is not necessary to run the `sdinst.exe` program that ships with the card because Lookout*Direct* downloads the KT-emulation module automatically as part of the initialization process. It is, however, necessary to tell Lookout*Direct* the port address specified by the switch settings on the card.



Name: AB1

PLC Address (0 - 77): 1 octal

Interface: S-S 5136-SD

PLC model: PLC5/15

1784-KT card settings

Card number ● 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8

Memory address: D400

Port: 250

Node address: 0 octal

□ Card exists in this computer

Baud rate: 57.6k

IRQ: (all cards) ○ 3 ○ 4 ○ 5 ○ 7 ○ 9 ○ 10 ○ 11 ○ 12 ○ 15 ● none

PollRate = 0:01

Poll =

Communication alarm priority: 8

Retry attempts: 4 Receive timeout: 500 msecs

Skip every 5 poll requests after comm failure

Allen-Bradley Register Addressing

Lookout*Direct* has adopted a sequential, flat addressing scheme for the Allen-Bradley PLCs. The addresses are sequential by data type, having nothing to do with the actual slot number. For example, consider the following slot configuration:

PLC slot Location	Type of Module in PLC Slot	Channel Address in Lookout*Direct*
slot 1	16 channel analog input	I:0-I:16
slot 2	16 channel analog output	O:0-O:16
slot 3	8 channel input, 8 channel output	I:17-I:24, O:17-O:24
slot 4	16 channel analog input	I:25-I:32

You can see from the example, the slot number of the module is irrelevant to Lookout*Direct*, and the input channels follow consecutive numbers as the output channels follow their own consecutive numbers.

Allen-Bradley Data Members

Each AB object contains a great deal of data. All readable and writable members (inputs/outputs) are bundled with the object. As soon as you create an AB object you immediately have access to all the object data members.

The AB object classes automatically generate an efficient read/write blocking scheme based on the inputs and outputs you are using in your process file. You are not required to build your own I/O blocking table. However, you can ensure peak performance by organizing your PLC data into contiguous groups.

Table 3-2. AB_PLC2 Data Members

Data Member	Type	Read	Write	Description
0 - 7777	numeric	yes	yes	16-bit signed binary word ranging from –32,768 to +32,767
0_0 - 7777_17	logical	yes	no	One bit within a 16-bit binary word
CommFail	logical	yes	no	Object-generated signal that is ON if, for any reason, Lookout*Direct* cannot communicate with the PLC.

Table 3-2. AB_PLC2 Data Members (Continued)

Data Member	Type	Read	Write	Description
OffHook	logical	no	yes	When true, instructs the PLC to retain exclusive use of its assigned communication port. This prevents Lookout*Direct* from hanging up between polls, saving the redial overhead. This also prevents other blocks from communicating over the same channel.
Poll	logical	no	yes	When transitioned from false to true, the Lookout*Direct* object polls the PLC device
PollRate	numeric	no	yes	Specifies the frequency at which the Lookout*Direct* object polls the PLC device
Update	logical	yes	no	Object-generated signal that pulses each time the object polls the device

Table 3-3. AB_SLC500 Data Members

Data Member	Type	Read	Write	Description
B:0 - B255:255	numeric	yes	yes	16-bit signed binary word ranging from –32,768 to +32,767
B:0_0-B255:255_15	logical	yes	yes	One bit within a 16-bit binary word
B_0 - B255_4095	logical	yes	yes	One bit within the specified datafile. For example, B3_32 specifies datafile 3, word 2, bit 1.
C:0.ACC - C255:255.ACC	numeric	yes	yes	Counter accumulated value. Two-byte signed word ranging from –32,768 to +32,767
C:0.PRE - C255:255.PRE	numeric	yes	yes	Preset counter value. Two-byte signed word ranging from –32,768 to +32,767
C:0_CU - C255:255_CU	logical	yes	yes	Counter up-enable bit.
C:0_DN - C:255:255_DN	logical	yes	yes	Counter done bit.

Table 3-3. AB_SLC500 Data Members (Continued)

Data Member	Type	Read	Write	Description
C:0_OV - C255:255_OV	logical	yes	yes	Counter overflow bit.
C:0_UA - C255:255_UA	logical	yes	yes	Counter update accumulation bit (HSC in fixed controller only)
C:0_UN - C255:255_UN	logical	yes	yes	Counter underflow bit.
C:0_CD -C255:255_CD	logical	yes	yes	Counter down-enable bit.
CommFail	logical	yes	no	Object-generated signal that is ON if, for any reason, Lookout*Direct* cannot communicate with the SLC.
F:0-F255:255	numeric	yes	yes	Floating point value
I:0 - I:30	numeric	yes	yes	Unsigned 16-bit input value ranging from 0 to 65,535
I:0_0 - I:30_15	logical	yes	yes	One bit within a 16-bit input word
N:0 - N255:255	numeric	yes	yes	16-bit signed integer value ranging from –32,768 to +32,767.
N:0_0 - N255:255_15	logical	yes	yes	One bit within a 16-bit signed integer word
O:0 - O:30	numeric	yes	yes	Unsigned 16-bit output value ranging from 0 to 65,535
O:0_0 - O:30_15	logical	yes	yes	One bit within a 16-bit output word
OffHook	logical	no	yes	When true, instructs the PLC to retain exclusive use of its assigned communication port. This prevents Lookout*Direct* from hanging up between polls, saving the redial overhead. This also prevents other blocks from communicating over the same channel.
Poll	logical	no	yes	When transitioned from false to true, the Lookout*Direct* object polls the SLC device

Table 3-3. AB_SLC500 Data Members (Continued)

Data Member	Type	Read	Write	Description
PollRate	numeric	no	yes	Specifies the frequency at which the Lookout*Direct* object polls the SLC device
R:0_FD - R255:255_FD	logical	yes	yes	Control “found” single-bit logical indicator
R:0.LEN - R255:255.LEN	numeric	yes	yes	Control “length” signed integer ranging from –32,768 to +32,767
R:0.POS - R255:255.POS	numeric	yes	yes	Control “position” signed integer ranging from –32,768 to +32,767
R:0_DN - R255:255_DN	logical	yes	yes	Control “done” single-bit logical indicator
R:0_EM - R255:255_EM	logical	yes	yes	Control “empty” single-bit logical indicator
R:0_EN -R255:255_EN	logical	yes	yes	Control “enable” single-bit logical indicator
R:0_ER - R255:255.ER	logical	yes	yes	Control “error” single-bit logical indicator
R:0_EU - R255:255_EU	logical	yes	yes	Control “enable unloading” single-bit
R:0_IN - R255:255.IN	logical	yes	yes	Control “inhibit comparison” flag single-bit logical indicator
R:0_UL - R255:255_UL	logical	yes	yes	Control “unload” single-bit logical indicator
S:0 - S:96	numeric	yes	yes	SLC status file containing a signed integer ranging from –32,768 to +32,767 (see Allen-Bradley documentation)
S:0_0 -S:96_15	logical	yes	yes	Individual SLC status bits (see Allen-Bradley documentation)
ST9:0 - ST255:255	text	yes	yes	String, limited to 83 characters in length. See the note on Allen-Bradley string data members at the end of this table.

Table 3-3. AB_SLC500 Data Members (Continued)

Data Member	Type	Read	Write	Description
T:0.ACC - T255:255.ACC	numeric	yes	yes	Accumulated timer value ranging from –32,768 to +32,767
T:0.PRE - T255:255.PRE	numeric	yes	yes	Preset timer value ranging from –32,768 to +32,767
T:0_DN - T255:255_DN	logical	yes	yes	Timer “done” single-bit logical indicator
T:0_EN - T255:255_EN	logical	yes	yes	Timer “enabled” single-bit logical indicator
T:0_TT - T255:255_TT	logical	yes	yes	Timer “timing” single-bit logical indicator
Update	logical	yes	no	Object-generated signal that pulses each time the object polls the device

Note There is no Allen-Bradley default file type associated with strings. You must configure your Allen-Bradley device to have a string file. See your Allen-Bradley documentation for details on this configuration procedure.

The string data member only works with the SLC 500 Enhanced series of PLCs, including the SLC 5/03; OS301 and SLC 5/04., and with the PLC 5 Enhanced series, PLC 5/11, PLC 5/20, PLC 5/30, PLC 5/40, PLC 5/60, PLC 5/80.

Table 3-4. AB_PLC5 Data Members

Data Member	Type	Read	Write	Description
B:0 - B999:999	numeric	yes	yes	16-bit signed binary word ranging from –32,768 to +32,767
B:0_0-B999:999_15	logical	yes	yes	One bit within a 16-bit binary word.
B_0 - For B999_15999	logical	yes	yes	One bit within the specified datafile; for example, B3_32 specifies datafile 3, word 2, bit 1.
C:0.ACC - C999:999.ACC	numeric	yes	yes	Counter accumulated value. Two-byte signed word ranging from –32,768 to +32,767
C:0.PRE - C999:999.PRE	numeric	yes	yes	Preset counter value ranging from –32,768 to +32,767.

Table 3-4. AB_PLC5 Data Members (Continued)

Data Member	Type	Read	Write	Description
C:0_CD - C999:999_CD	logical	yes	yes	Counter down-enable bit.
C:0_CU - C999:999_CU	logical	yes	yes	Counter up-enable bit.
C:0_DN - C:999:999_DN	logical	yes	yes	Counter done bit.
C:0_OV - C999:999_OV	logical	yes	yes	Counter overflow bit.
C:0_UA - C999:999_UA	logical	yes	yes	Counter update accumulation bit (HSC in fixed controller only).
C:0_UN - C999:999_UN	logical	yes	yes	Counter underflow bit.
CommFail	logical	yes	no	Object-generated signal that is ON if, for whatever reason, Lookout*Direct* cannot communicate with the PLC.
F:0-F999:999	numeric	yes	yes	Floating point value.
I:0 - I:277	numeric	yes	yes	Unsigned 16-bit input value ranging from 0 to 65,535.
I:0_0 - I:277_17	logical	yes	yes	One bit within a 16-bit input word (octal).
N:0 - N999:999	numeric	yes	yes	16-bit signed integer value ranging from –32,768 to +32,767.
N:0_0 - N999:999_15	logical	yes	yes	One bit within a 16-bit signed integer word.
O:0 - O:277	numeric	yes	yes	Unsigned 16-bit output value ranging from0 to 65,535.
O:0_0 - O:277_17	logical	yes	yes	One bit within a 16-bit output word (octal).

Table 3-4. AB_PLC5 Data Members (Continued)

Data Member	Type	Read	Write	Description
OffHook	logical	no	yes	When true, instructs the PLC to retain exclusive use of its assigned communication port. This prevents Lookout*Direct* from hanging up between polls, saving the redial overhead. This also prevents other blocks from communicating over the same channel.
Poll	logical	no	yes	When transitioned from false to true, the Lookout*Direct* object polls the PLC device.
PollRate	numeric	no	yes	Specifies the frequency at which the Lookout*Direct* object polls the PLC device.
R:0.LEN - R999:999.LEN	numeric	yes	yes	Control “length” signed integer ranging from –32,768 to +32,767.
R:0.POS - R999:999.POS	numeric	yes	yes	Control “position” signed integer ranging from –32,768 to +32,767.
R:0_DN - R999:999_DN	logical	yes	yes	Control “done” single-bit logical indicator.
R:0_EM - R999:999_EM	logical	yes	yes	Control “empty” single-bit logical indicator.
R:0_EN - R999:999_EN	logical	yes	yes	Control “enable” single-bit logical indicator.
R:0_ER - R999:999.ER	logical	yes	yes	Control “error” single-bit logical indicator.
R:0_EU - R999:999_EU	logical	yes	yes	Control “enable unloading” single-bit logical indicator.
R:0_FD - R999:999_FD	logical	yes	yes	Control “found” single-bit logical indicator.
R:0_IN - R999:999.IN	logical	yes	yes	Control “inhibit comparison” flag logical indicator.

Table 3-4. AB_PLC5 Data Members (Continued)

Data Member	Type	Read	Write	Description
R:0_UL - R999:999_UL	logical	yes	yes	Control “unload” single-bit logical indicator.
S:0 - S:127	numeric	yes	yes	PLC status file containing a signed integer ranging from –32,768 to +32,767 (see Allen-Bradley documentation).
S:0_0 - S:127_15	logical	yes	yes	Individual PLC status bits (see Allen-Bradley documentation).
ST9:0 - ST255:255	text	yes	yes	String, limited to 83 characters in length. See the note on Allen-Bradley string data members at the end of this table.
T:0.ACC - T999:999.ACC	numeric	yes	yes	Accumulated timer value ranging from –32,768 to +32,767.
T:0.PRE - T999:999.PRE	numeric	yes	yes	Preset timer value ranging from –32,768 to +32,767.
T:0_DN - T999:999_DN	logical	yes	yes	Timer “done” single-bit logical indicator.
T:0_EN - T999:999_EN	logical	yes	yes	Timer “enabled” single-bit logical indicator.
T:0_TT - T999:999_TT	logical	yes	yes	Timer “timing” single-bit logical indicator.
Update	logical	yes	no	Object-generated signal that pulses each time the object polls the PLC device.

Note There is no Allen-Bradley default file type associated with strings. You must configure your Allen-Bradley device to have a string file. See your Allen-Bradley documentation for details on this configuration procedure.

The string data member only works with the SLC 500 Enhanced series of PLCs, including the SLC 5/03; OS301 and SLC 5/04., and with the PLC 5 Enhanced series, PLC 5/11, PLC 5/20, PLC 5/30, PLC 5/40, PLC 5/60, PLC 5/80.

Allen-Bradley Error Messages

AB objects report the statuses of commands they issue to AB devices. When Lookout*Direct* receives a response from an AB device, it reads the status (STS) byte and, if necessary, the extended status (EXT STS) byte to verify the device executed the Lookout*Direct* command properly. If the command was not executed properly, Lookout*Direct* reports the failure as an alarm containing the status code and its meaning. The following is an example of such an alarm:

```
EXT STS = 0F: not enough levels in address
```

AB object classes can also generate alarms internally. The following is a list of AB alarms generated by Lookout*Direct*, their descriptions, and possible responses. In the messages, *KT* is used to refer to any of the DH+ interface cards (1784-KT, 1784-KTx, 1784-PCMK, or 5136-SD) and *SS* is used to refer to the 5136-SD card.

Cannot resolve ip address: *address*

The AB object failed to find any node on the network that corresponds to the given IP address. Confirm that the IP address entered in the **Modify Object** dialog box is correct.

Cannot get session id from plc

The AB object sent a message to the PLC requesting a TCP/IP session and failed to receive a satisfactory response.

Cannot communicate with device (code=dd)

The AB object timed out while waiting for a response (via TCP/IP) from the PLC. If the code is 0, the object timed out while trying to establish the TCP/IP connection; if the code is 1, the object timed out while waiting for a session id from the PLC; if the code is 2, the object timed out while waiting for a response to a poll request. Confirm that the IP address of the PLC has been entered correctly and that the PLC is reachable over the TCP/IP network.

Download of file sdipds.ss1 failed

Lookout*Direct* was unable to write the KT-emulation program file to the 5136-SD card physical memory. This could be due to either an invalid port or memory address or to a faulty or improperly seated card.

Invalid memory address for card: *0xAAAA*

The memory address specified for the card (for example, D400) is not valid for this model of card. The address must be a multiple of 0x0100 and lie in the range 0xA000 to 0xDF00. Moreover, the KT, KTx, PCMK, and

5136-SD cards support different sets of valid memory addresses. See the documentation that shipped with the card for details.

Invalid node address for card: ***xx***

Node addresses must be between 0 and 63 decimal.

Invalid port number for SS card: ***0xPPP***

The port number specified for the 5136-SD card is invalid. See the documentation that shipped with the card for the list of valid port addresses.

Invalid port or memory address

Lookout*Direct* was unable to write to the 5136-SD card physical memory. This could be due to either an invalid port or memory address or to a faulty or improperly seated card.

KT card failed to find resources
KT card receive mailbox is in an invalid state
KT card send mailbox is in an invalid state

You will probably never see one of these alarms. If you do, call National Instruments and ask for technical support.

KT card dual-ported memory test failed at location ***xxxx***

The interface card failed a memory test when it was first powered on. The memory test reads, writes and rereads the dual-ported memory to ensure memory access by the card. Verify that the card is configured for the memory address that you specified. Verify that your memory manager (like EMM386) excludes the appropriate portion of memory. Verify that your card is not trying to use the same memory location as another card. You may need to restart the card by calling up the AB object definition dialog box and selecting **OK**. If that does not work try rebooting the computer. Other causes can include memory conflicts, a bad interface card, or a misbehaving driver.

KT card CTC test timed out
KT card CTC test failed with status code ***xx***

The interface card failed the Counter Timer Circuit test when it was first powered on. This test verifies proper functionality of the card timer and counter modes over all CTC channels. You may need to restart, reseat, or replace the interface card.

KT card timed out while loading protocol code
KT card failed with status code *xx* while loading protocol

Lookout*Direct* was not able to transfer a loader file to the card and subsequently download the card protocol firmware. Try to restart the interface card by calling up the AB object definition dialog box and selecting **OK**. If that does not work try rebooting the computer.

KT card is no longer responding

The Lookout*Direct* AB object did not receive the interface card heartbeat within the last second. Normally, the card generates a heartbeat any time it receives the DH+ network token. If the alarm does not deactivate after 30 seconds, try to restart the card by calling up the AB object definition dialog box and selecting **OK**. If that does not work try restarting Lookout*Direct* or rebooting the computer.

KT card memory address conflicts with card *n*

Lookout*Direct* found another interface card with the same memory address. Be sure that the memory address on each interface card is different and that the corresponding **Memory address** in the Lookout*Direct* object matches the card address.

KT card not present in this computer

The **Card exists in this computer** check box is deselected (this is the default setting). Select **Object»Modify** to retrieve the PLC definition parameters dialog box and select the **Card exists in this computer** check box and **OK** to initialize the card.

KT card RAM test timed out
KT card RAM test failed with status code *xx*

The interface card failed a memory test when it was first powered on. The memory test writes a pattern to on-board RAM and reads its content to verify the card memory is working. Confirm that the memory address specified in the **Object»Modify** dialog box is correct. You may need to restart, reseat, or replace the card.

KT card signature test failed

The AB object does not recognize the card. Make sure that the interface card is actually installed in the PC and that you indicated the correct memory address in Lookout*Direct*. You may need to reseat the card, or the card may require repair. Also ensure that you did not identify more **Card Numbers** than actual physical cards.

KT card SI0 test timed out
KT card SI0 test failed with status code *xx*

The interface card failed the Serial Input Output test when it was first powered on. You may need to restart, reseat, or replace the interface card.

KT card send mailbox timed out

The AB object timed out while waiting for the KT card to signal that it is ready to be given a new message to send. This is most likely due to a communication problem between the computer and the PLC. Confirm that the network cable is properly installed and that the PLC is turned on.

NAK response received

The AB object received a NAK (not acknowledged) response when it polled the device. The device received a command from Lookout*Direct* but it did not accept the message. The command that the device received may be incomplete or contain irregularities due to poor network performance. If your Serial Port is configured for radio, you may need to adjust the **RTS delay off** setting. Also consider increasing the number of **Retry attempts**.

No response — no ACK for our transmission

Lookout*Direct* is not getting any response from the device. This could be caused by just about anything. Verify that the **Data rate**, **Parity**, and **Error detection** settings are the same as the settings on the device. Make sure you are using the proper **Serial port** on your computer. Verify that the device interface module and other network equipment is connected and working properly. If you are using a modem, verify that your object **Phone number** and the serial port **Dial-up** settings are correct. This may also be caused by low level noise or reflections on the highway, or marginal circuitry on a card.

No response within timeout period
No response received after receiving ENQ

The AB object received an acknowledgment of its poll from the device. The device accepted the command from Lookout*Direct*. However, the device did not appear to send anything else back in response. You may have to increase **Receive timeout** to make sure Lookout*Direct* allows enough time to receive the message.

EOT response received

The AB object received an EOT (end of transmission) response when it polled the device indicating that the device did not have a message ready to give in response to the Lookout*Direct* poll request. It is unlikely that you will ever see this error message.

Received TSN does not match
Response message garbled -- bad CRC or bad BCC
Response message garbled -- no DLE EXT
Response message garbled -- bad DLE follower

The AB object is receiving messages from the device. However, the messages may be failing the selected data integrity test. Verify that the object **Error detection** setting is the same as the settings on the device. Another cause may be that the last part of the message is actually getting clipped off before it is completed. You may have to increase the **Receive gap** Serial Port setting to ensure Lookout*Direct* is receiving the entire message. If your Serial Port is configured for radio this could be caused by an audible squelch tail occurring at the end of a radio transmission. You may need to adjust the **RTS delay off** or the **CTS timeout** settings. Also consider increasing the number of **Retry attempts**.

Socket communications error *dd: msg*

The AB object has encountered a problem while attempting to communicate using TCP/IP. The error number *dd* and corresponding error message *msg* give further information. Confirm that the IP address of the PLC has been entered correctly and that the PLC is reachable over the TCP/IP network.

SS card failed

This message is suffixed with an error message read from the card itself. You may need to contact the vendor of your card for technical support.

SS card failed while performing diagnostics

Lookout*Direct* successfully wrote the KT-emulation program to the 5136-SD card, but the program failed to terminate. Try running the `sdinst.exe` program that ships with the 5136-SD card, using the CHK option to confirm that the card is working properly.

Unable to access physical memory at segment *0xAAAA*

Lookout*Direct* was unable to access the memory at the given segment address. The memory may already be in use by the operating system or by another application. Either change the object memory address (which may involve changing switch settings on the card itself) or, if you are using a memory manager, make sure that it is excluding the correct portion of memory.

Unable to open port *0xPPP* for SS card

Lookout*Direct* was unable to open the port number specified for the 5136-SD card. Make sure that you have specified the port that is selected by the jumper settings on the card. Make sure that port and the following two ports (for example, 250, 251, and 252) are not in use by any other devices in your computer.

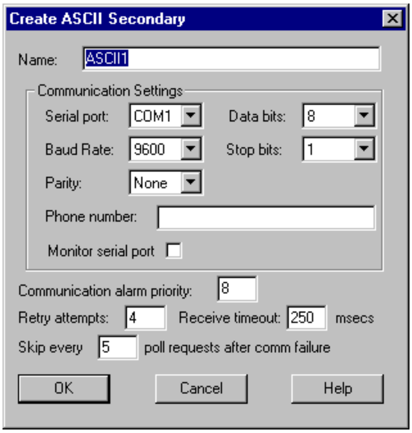
Unexpected data response length
Unexpected response
Unhandled error

You will probably never see one of these alarms. If you do, call National Instruments and ask for technical support.

ASCII

ASCII is a protocol driver class Lookout*Direct* uses to communicate with any serial device that accepts ASCII characters. This object is only available with 32-bit versions of Lookout*Direct*.

An ASCII object contains no predefined data points. When you create an ASCII object, you must define your data request strings as well as the template Lookout*Direct* uses to parse the response frame.



Create ASCII Secondary

Name: ASCII1

Communication Settings

Serial port: COM1

Data bits: 8

Baud Rate: 9600

Stop bits: 1

Parity: None

Phone number:

Monitor serial port □

Communication alarm priority: 8

Retry attempts: 4

Receive timeout: 250 msecs

Skip every 5 poll requests after comm failure

Serial port specifies which COM port the object uses for communicating to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports…** command.

Baud rate indicates the rate that Lookout*Direct* uses to communicate with the hardware device.

Data bits indicates the number of data bits that Lookout*Direct* uses to communicate with the hardware. This setting should match the selection made on the physical device.

Stop bits indicates the number of stop bits that Lookout*Direct* uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Parity indicates the parity that Lookout*Direct* uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Phone number specifies the number to be dialed if the selected serial port is configured for dial-up. This number only applies to the individual protocol object.

Monitor Serial Port specifies whether you can receive unsolicited frames.

Communication alarm priority determines the priority level of alarms generated by the ASCII object. Such alarms are typically related to communications with the physical device.

Retry attempts specifies the number of times Lookout*Direct* attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the ASCII object generates an alarm and releases the communication port back to the communications service which then moves on to the next device in the polling queue (if any). Refer to Chapter 3, *Serial Communications*, in the *LookoutDirect Developer's Manual* for more information.

Receive timeout is the amount of time Lookout*Direct* waits for a response from a device before retrying the request.

The **Skip every *N* poll requests after comm failure** setting instructs Lookout*Direct* not to poll a device it has lost communication with on every scheduled poll. Instead, Lookout*Direct* polls the device only once in the specified number of poll cycles. Once communication has been reestablished, the device is polled on its regular cycle.

ASCII Data Members

Table 3-5. ASCII Data Members

Data Member	Type	Read	Write	Description
CommFail	logical	yes	no	Object-generated signal that is on if Lookout*Direct* cannot communicate with the device(s).
OffHook	logical	no	yes	Keeps the driver from releasing the serial port.
Request	text	yes	no	Exact request frame sent.
RequestFormat	text	no	yes	Format used to create request frame.
Response	text	yes	no	Exact response frame received.
ResponseFormat	text	no	yes	Format used to parse response frame.

Table 3-5. ASCII Data Members (Continued)

Data Member	Type	Read	Write	Description
RQSum:1:1 - RQSum255:255	numeric	yes	no	Request byte sum
RQV1, RQV512	numeric	no	yes	Variable list used to populate request frame with numeric values.
RQV1.logical, RQV512.logical	logical	no	yes	Variable list used to populate request frame with logical values.
RQV1.txt, RQV512.txt	text	no	yes	Variable list used to populate request frame with text values.
RSFilter	text	no	yes	All characters in this string will be filtered out of the incoming response before processing.
RSSum1:1 - RSSum255:255	numeric	yes	no	Response byte sum
RSV1, RSV512	numeric	yes	no	Variable list used to store values retrieved from response frame.
RSV1.logical, RSV512.logical	logical	yes	no	Variable list used to store values retrieved from response frame.
RSV1.txt, RSV512.txt	text	yes	no	Variable list used to store values retrieved from response frame.
Send	logical	no	yes	Sends request frame.
Update	logical	yes	no	Object-generated signal that pulses low each time it polls the device.
RSVn, RSVn.txt and RSVn.logical all represent the same value in different forms RQVn, RQVn.txt and RQVn.logical all represent the same value in different forms				

Request and Response Format Strings

The request and response format strings consist of static characters and markers that control how the request and response frames respectively are formatted or decoded. The request format string is used to *create* the request frame, which is sent to the device, while the response format string is used to *decode* the response frame, which comes from the device.

Static characters in the format strings are reproduced exactly in the request or response frame. Markers specify the location within the frame and type of data which should be found there, such as five characters read as an unsigned integer, for example. The ASCII object constructs a request frame by processing the sequence of static characters and markers in the request format string, and including data from RQV data members.

The response format string decodes a response frame using an analogous process, storing the results in RSV data members.

To construct a request frame, the ASCII object parses the request format string character by character. Static characters are copied directly to the request frame. When a marker is encountered the ASCII object reads a value from the appropriate RQV variable and places it into the request frame.

There are 512 RQV and RSV values provided for in the ASCII object data member collection. The first marker in a format string uses the value from RQV1 (or RQV1.txt or RQV1.logical), the next marker uses the value RQV2, and so on. Values taken from Response strings are stored in RSV data members in the same way.

Keep in mind that writing into RQV1 changes the value both for RQV1.text and RQV1.logical. Their only difference is the format in which they are represented. The same principle applies to the RSV data members.

Note There is no precedence to the order in which multiple objects connected to the same variable number initialize upon opening the process file. Consider, for example, the case in which a Pot object is connected to RQV1 while a TextEntry object is connected to RQV1.txt. You should take care to initialize such variables to the proper value after opening a process file.

To decode a response frame, the ASCII object compares the response frame to the response format string character by character. The static characters in the response frame must match those in the response format string or the decoding process terminates. Static characters are, in effect, discarded by the ASCII object as they are matched between the response format string and the response frame.

When the ASCII object encounters a marker, it places the data indicated by the marker into the appropriate RSV data member.

The conversion of a portion of the response frame to a data type specified by a marker in the response format string must be valid, or the process will terminate.

If nothing halts the process, decoding terminates when the end of the response frame string is reached.

There are examples of both request frames and response frames at the end of this section, but for the examples to make sense, you must first understand the ASCII object markers.

ACSII Object Markers

The general format for a marker is:

%[width][type]

Each field in the marker format is a single character or a number signifying a particular format option.

The **%** sign denotes the beginning of the marker. For example, to specify that a percent-sign character is a static character part of the frame, use %%.

Width is a positive decimal integer specifying the number of characters that particular value occupies in the frame. By default ASCII pads the value with blank spaces if the value takes up fewer characters than the value specified by width. Including a 0 before the width value forces the ASCII object to pad with zeroes instead of blank spaces.

Type determines whether the field is interpreted as a character, a string, or a number.

Table 3-6. Data Types Allowed by ASCII

Character	Data Type
c	Character
d	Decimal integer
O, o	Octal
x, X	Hexadecimal integer
u	Unsigned decimal integer
e, f	Floating-point
s	String

Table 3-6. Data Types Allowed by ASCII

<table>
<tr><td>b*</td><td>Byte (binary)</td></tr>
<tr><td colspan="2">*For the %b data type:
– Number of bytes can be specified, for example %3b, %2b.
– Response format can read as either signed or unsigned, for example %^b and %3^b are signed and %b is unsigned.
– Endian order can be specified, for example %3~b is big endian and %3b is little endian. %5~^b and %2^~b forms are also valid.</td></tr>
</table>

The simplest format specification contains only the percent sign and a type character (for example, %s). That would place the value in the response frame in the RSV1.txt data member.

Request Format String	RQV1	Request Frame
>%5d	34	> 34
>%05d	34	>00034

The request format string also has a precision value in the form **%[width].[precision][type]**. This specifies the number of digits to the right of the decimal point, if any, in the request frame. If you use a float (%f) and do not specify a precision value, the ASCII object assumes a default of 6.

Characters are converted and stored in RSV data members from response frames in the order they are encountered in the response format. However, fewer than **[width]** characters may be read if a white-space character (space, tab, or newline) or a character that cannot be converted according to the given format occurs before **[width]** is reached.

Values needed for request frames come from the RQV data members, and are also used in the order in which they occur in the request format.

To read strings not delimited by space characters, or that contain spaces, you can substitute a set of characters in brackets (`[ ]`) **s** (string) type character. The corresponding input field is read up to the first character that does not appear in the bracketed character set. Using a caret (^) as the first character in the set reverses this effect: the ASCII object reads input field up to the first character that does appear in the rest of the character set.

Response Format String	**RSV1.txt**	**Response Frame**
$%[A – Z,a – z,]$	Natl Inst	$Natl Inst$
>%[^,s]	days	>day

Notice that %[a – z] and %[z – a] are interpreted as equivalent to %[abcde…z], and that the character set is case sensitive. Valid control characters accepted include \a, \A, \b, \B, \f, \F, \n, \N, \r, \R, \t, \T, \v, \V, \\, \’, \”, and \?.

Any ASCII character can be specified in the format string using \xbb or \nnn masks. \xbb is a hexadecimal byte with each b representing a valid hex character in the range (0...9, a...F), for example \xff or \x1a.

\nnn is an octal byte with each n being a valid octal character in the range 0 to 7, for example \123 or \347. THis value may not exceed 255 as it is meant to represent a single byte of data. These two features can be used to create and compare static characters. These can also be specified in the regular expression, for example %[\xff, \123, \a, \b] is a valid frame.

Note The brackets only work in response format strings. They have no effect in the request format string.

The ASCII object scans each field in the response frame character by character. It may stop reading a particular field before it reaches a character for a variety of reasons:

- The specified width has been reached.
- The next character cannot be converted as specified.
- The next character conflicts with a character in the response format string that it is supposed to match.
- The next character fails to appear in a given character set.

No matter what the reason, when the ASCII object stops reading a field, the next field is considered to begin at the first unread character. The conflicting character, if there is one, is considered unread and is the first character of the next field.

Entering ASCII Object Format String

For a static connection to one of the format data members, enter your format string in the yellow field box in the Edit Connections dialog box. Remember to begin and end the format strings with quotation marks so that Lookout*Direct* accepts the string input.

You can also connect any valid text data member, such as a text entry object, to the format data members.

Request Frame Construction Examples

Request Format String	RQV	Request Frame

<01%4u%s	RQV1=1234
	RQV2.txt=Steph
		<011234Steph

<01%04u%s	RQV1=34
	RQV2.txt=Steph
		<010034Steph

<01% 4u%s	RQV1=34
	RQV2.txt=Steph
		<01 34Steph

A zero in front of the four pads with zeroes; a space pads with spaces.

Response Format Examples

Response Frame	Response Format String	RSV
*(16.38:	*(%52f:	RSV1=16.38

Note The decimal point counts as a character when decoding floats (%f). Also, decimal points denoting precision are not allowed when decoding a float in the response frame.

Response Frame	Response Format String	RSV
>>Test Text<<	>>%s<<	RSV1.txt=Test

The space between the words terminates the conversion. See the preceding bracketed character example in order to span a space or other special characters.

Response Frame	Response Format String	RSV

>>Test Text<<	>>%s%s<<	RSV1.txt=Test
		RSV2.txt=Text

>>DogCat<<	>>%3s%3s<<	RSV1.txt=Dog
		RSV2.txt=Cat

The response format uses a space as a delimiter.

Using Sum Data Members

The ASCII object includes summing data members you can use to calculate checksum characters. This can be a checksum you want to write into an outgoing request frame or a checksum you want to verify in an incoming response frame.

For example, if you want to calculate a checksum for the request `A00B`, you would use an `RQSum` (request sum) data member. In the case of `A00B`, you would use `RQSum1:4`, which would give you a sum of the ASCII byte values of characters 1 through 4. Once you have this sum, you can manipulate it mathematically any way necessary for the checksum value you need. You can then insert this value at the end of your frame as a byte (`%b`) or a series of bytes.

The same technique works in reverse for `RSSum` (response sum) data members.

For example, consider the response `Z00A@`. You know that you are expecting 4 bytes plus a checksum. Assuming that this checksum calculation involves the first four characters, use `RSSum1:4` to get the byte sum of characters 1 through 4. After performing the appropriate mathematical manipulation, you can compare this value with the actual byte read from the frame, and determine when there is a checksum failure.

Note There are many different methods for calculating checksums, and these data members cannot support all of them. Before attempting to use them for checksum calculation, make sure your checksum can be calculated from a simple byte sum of characters in the frame.

ASCII Error Messages

No response from device within timeout period

Lookout*Direct* received no response from the device within the **Receive timeout** period. The ASCII object was able to establish a socket, but when it sent a message to the device, the device did not respond—as if it were not there. You may have to significantly increase **Receive timeout** (and **Poll Rate**) to ensure Lookout*Direct* is allowing enough time to receive the expected response. Also, verify your cable connections, power, configuration settings, and IP settings.

Not enough data to send a valid frame

This means that the ASCII object has not received enough data to fill in all the variables in the Request Format frame. This could mean that you do not have connections made to all of the RQVs that the ASCII object is expecting.

Frame Error (garbled)

ASCII got a response frame, but static characters in the response did not match up to the response format string.

Data type or length does not match format string

ASCII got a response frame, but certain characters in the response were not in the format stated by the markers in the response format string.

Illegal control character

The request frame had a % modifier followed by an invalid control code.

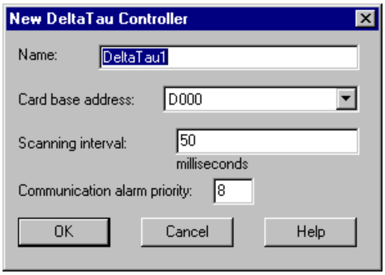
Related Objects *IPASCII*

DeltaTau

DeltaTau is a protocol driver object class Lookout*Direct* uses to communicate with Delta Tau Data Systems PMAC Motion Controllers. Create a DeltaTau object for each card installed in the computer.

This object class communicates with Delta Tau PMAC cards through dual-ported memory, so be sure that your PMAC hardware includes the dual-ported RAM option.

The following figure shows a Delta Tau card configured to use PC memory beginning at address D000.


New DeltaTau Controller

Name: DeltaTau1

Card base address: D000

Scanning interval: 50 milliseconds

Communication alarm priority: 8

Card base address specifies the beginning memory location of the dual ported RAM address. It should match card settings.

Scanning Interval identifies the frequency that the DeltaTau object in Lookout*Direct* polls the PMAC Motion Controller. Intervals can range from 10 ms to 1,000 ms.

Communication alarm priority specifies the priority level of alarms generated by the object.

DeltaTau Data members

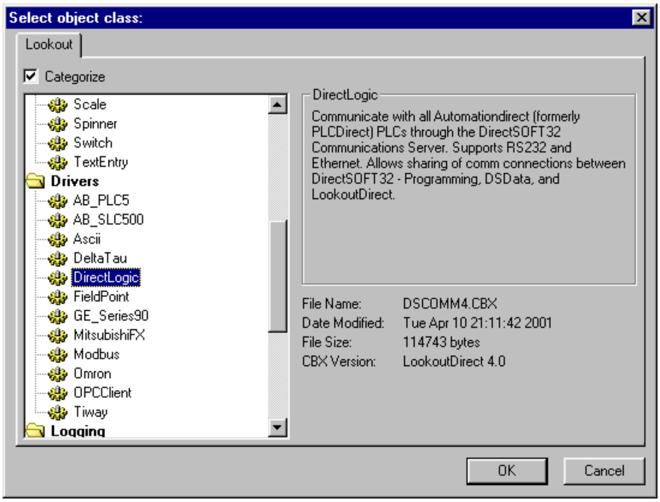
Like other protocol driver objects, DeltaTau objects contain a great deal of data. All readable and writable members (inputs/outputs), polling instructions, and so on, are bundled with the object. Therefore, as soon as you create a DeltaTau object you immediately have access to all the object data members. The following table lists data members for the DeltaTau object.

Table 3-7. DeltaTau Data Members

Data Member	Type	Read	Write	Description
DS0 – DS8190	numeric	yes	yes	Signed 32-bit word ranging from –2,147,483,648 to 2,147,483,647
DW0 – DW8190	numeric	yes	yes	32-Bit double-precision word ranging from 0 to 4,294,967,295
F0 – F8188	numeric	yes	yes	32-bit IEEE floating point word
S0 – S8190	numeric	yes	yes	Signed 16-bit word ranging from –32,768 to 32,767
Update	logical	yes	no	Driver-generated signal that pulses each time Lookout*Direct* scans the PMAC Motion Controller card
W0 – W8190	numeric	yes	yes	16-Bit word ranging from 0 to 65,535
W0.0 – W8190.15	logical	yes	yes	1 Bit in a 16-Bit word

DirectLogic

The DirectLogic protocol driver object class is used to communicate to PLC Direct by Koyo 105, 205, 305, and 405 PLCs. This driver object supports additional Koyo private labeled PLCs under GE Series One, Texas Instruments Series 305, 405 and Siemens Simatic TI305, TI405 names. Please see the Protocol Reference Chart within the LookoutDirect Learning Guide to verify the part numbers of these additional Koyo PLCs. LookoutDirect supports both point-to-point, multidrop and Ethernet configurations. You can connect LookoutDirect to a fixed PLC Programming port, a networking DirectNET/CCM port, PLC DCM (data communications module), or to an Ethernet (ECOM) communications module.

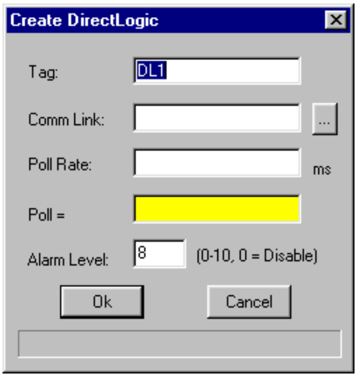


DirectLogic

Communicate with all Automationdirect (formerly PLCDirect) PLCs through the DirectSOFT32 Communications Server. Supports RS232 and Ethernet. Allows sharing of comm connections between DirectSOFT32 - Programming, DSData, and LookoutDirect.

File Name: DSCOMM4.CBX
Date Modified: Tue Apr 10 21:11:42 2001
File Size: 114743 bytes
CBX Version: LookoutDirect 4.0

After the DirectLogic driver class is selected the following dialog appears.



Tag: DL1

Alarm Level: 8 (0-10, 0 = Disable)

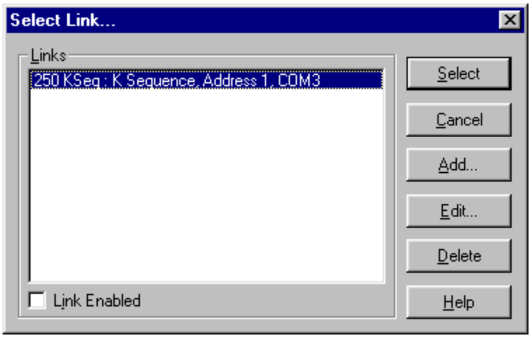
The selections are as follows:

Tag: User defined object name that can be referenced for connecting other objects.

Comm Link: The associated link chosen for the configured driver object. When the "…" button is pressed the **Select Link** dialog will appear.

Poll Rate= is a numerical expression that determines how often to poll the device in milliseconds.

Poll is a logical expression. When the expression changes from false to true, LookoutDirect polls the device one time. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.



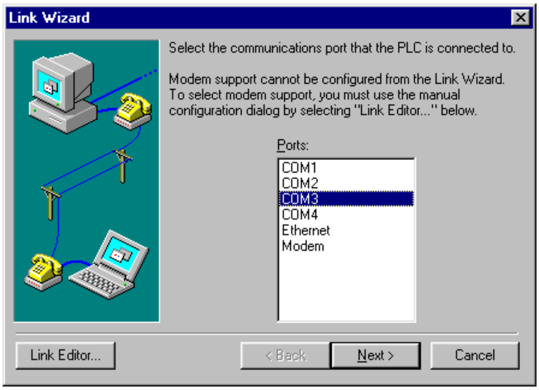
Links

250 KSeq : K Sequence, Address 1, COM3

□ Link Enabled

If a link is not available, you will have to add one. Click the **Add…** button to create a new communications link then select the correct Serial or Ethernet port that you have the cable attached from your PC to the PLC.

Note NOTE: If you are creating a serial Link that will connect through a modem, see the Modem Setup Guide Appendix C in the LookoutDirect Learning Guide. If you are creating an Ethernet Link, refer to the DirectSoft Communications Server online help.



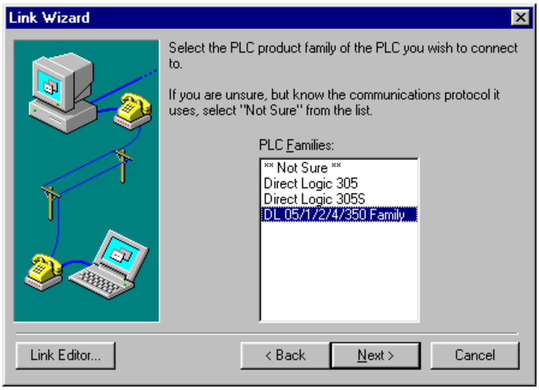
Link Wizard

Select the communications port that the PLC is connected to.

Modem support cannot be configured from the Link Wizard. To select modem support, you must use the manual configuration dialog by selecting "Link Editor..." below.

Ports:
COM1
COM2
COM3
COM4
Ethernet
Modem

After making your choice click on the **Next** button.



Link Wizard

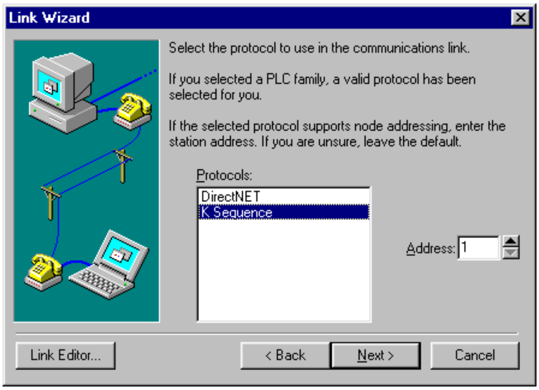
Select the PLC product family of the PLC you wish to connect to.

If you are unsure, but know the communications protocol it uses, select "Not Sure" from the list.

PLC Families:
** Not Sure **
Direct Logic 305
Direct Logic 305S
DL 05/1/2/4/350 Family

Select the PLC family by single clicking on the appropriate choice. If you are unsure of the PLC family but know which communications protocol to use, you can select the “ **Not Sure**” choice. If you are using a PLC Direct compatible PLC the LinkWizard can try to detect the model automatically.

Click on the Next> button when you are finished with your selection.



Link Wizard

Select the protocol to use in the communications link.

If you selected a PLC family, a valid protocol has been selected for you.

If the selected protocol supports node addressing, enter the station address. If you are unsure, leave the default.

Protocols:
- DirectNET
- K Sequence

Address: 1

Select either the DirectNET or K-Sequence protocol. If during the previous step you selected one of the families listed, the highlight bar will be on a valid protocol for that family of PLC. The choice of protocol to use will depend on two factors:

- Whether or not the PLC you are connected to supports the protocol on the port where the cable is connected. For a listing of protocols available on each port of DirectLogic compatible PLCs see the Protocol Reference Chart in the Creating and Managing Communication Links chapter within the LookoutDirect Learning Guide. This chart is also available in the DirectSoft Communications Server Help that can be accessed by clicking on the Link Editor button the clicking on the Help button.

- If you need to perform write operations to individual Discrete I/O points or control relays, then you must select K-sequence protocol. DirectNET protocol cannot write to individual bit locations.

If the PLC has been configured to a node Address other than 1, enter that address now. Click on the **Next** button when you are finished.

The DirectLogic driver object will now attempt to establish a communication Link with the PLC using the node address and protocol you have selected. It will try the combination of 9600 Baud, and Odd Parity first. If this combination is unsuccessful, an ‘auto-bauding’ sequence will be used to try and determine the correct baud rate and parity combination. If these attempts are unsuccessful, the following dialog is displayed. You can click the Link Editor button and manually attempt to adjust the port

configuration, or you can consult the Appendix B Communications Troubleshooting Guide
in the LookoutDirect Learning Guide.

DirectLogic Protocol Status

The DirectLogic driver object monitors protocol statistics and the data can be viewed by selecting Options >> DirectSoft Link Status. This menu selection will only be visible in the Options menu if a DirectLogic driver object was previously created in your lookout process. Select the correct object from the list that appears, and then click **Select**. The **Link Info** dialog box appears.

The **Link Info** dialog box is useful for troubleshooting communication errors. Each error is time stamped and has an error name and an extended error description. At anytime you can press the edit button to manually change a setting on your link or to press Auto then Accept to clear the error log. The Link Enable box allows you to turn the communications link On and Off.

105/205/350/405 DirectLogic PLC Family Data Members

Table 3-8. 105/205/350/405 DirectLogic PLC Data Members

Data Member	Type	Read	Write	Description
Activated	Logical	Yes	No	Object-generated signal when TRUE, this flag signifies an active5 communication connection between the process file and the PLC.
C0 – C3777	Logical	Yes	Yes	Control Relays – addressed in octal and mapped to V40600 – V40777.
CT0 – CT377	Logical	Yes	Yes	Counter status (done) bits – addressed in octal and mapped to V41140 – V41157
CTA0:B – CTA377:B	Numeric	Yes	Yes	Counter current value words BCD – addressed in octal and mapped to V01000 – V01377

Table 3-8. 105/205/350/405 DirectLogic PLC Data Members

CTA0:DB – CTA376:DB	Numeric	Yes	Yes	Counter current value double words (two adjacent addresses; 32-Bit) BCD – addressed in octal and mapped to V01000 – V01377
Failed	Logical	Yes	No	Object-generated signal when TRUE, this flag signifies the process file is no longer communicating with the PLC.
GX0 – GX3777	Logical	Yes	Yes	Remote I/O Inputs – addressed in octal and mapped to V40000 – V40177
GY0– GY3777	Logical	Yes	Yes	Remote I/O Outputs – addressed in octal and mapped to V40200 – V40277
Paused	Logical	Yes	No	Object-generated signal when TRUE, this flag signifies that the communication connection has paused but is still active and will go FALSE when its paused condition is satisfied. Usually caused by modifying the Link while it is in use.
S0 – S1777	Logical	Yes	Yes	Stage status (active) bits – addressed in octal and mapped to V41000 – V41077
SP0 – SP777	Logical	Yes	No	Special Relays (system status bits) – addressed in octal and mapped to V41200 – V41237
T0 – T0377	Logical	Yes	Yes	Timer status (done) bits – addressed in octal and mapped to V41100 – V41117
TA0:B – TA377:B	Numeric	Yes	Yes	Timer current value words BCD – addressed in octal and mapped to V00000 – V00377
TA0:DB – TA376:DB	Numeric	Yes	Yes	Timer current value double words (two adjacent addresses; 32-Bit) BCD – addressed in octal and mapped to V00000 – V00377
V0 – V41237	Numeric	Yes	Yes	Single (16-Bit) V-memory registers decimal

Table 3-8. 105/205/350/405 DirectLogic PLC Data Members

V0:D – V41237	Numeric	Yes	Yes	Double (two adjacent registers; 32-Bit) V-memory registers decimal
V0:B – V41237:B	Numeric	Yes	Yes	Single (16-Bit) V-memory registers BCD 0- 9999
V0:DB – V41236:DB	Numeric	Yes	Yes	Double (two adjacent registers; 32-Bit) V-memory registers BCD 0- 99999999
V0:R - V41236:R	Numeric	Yes	Yes	Double word V-memory registers signed real (IEEE 32-Bit Floating Point)
V0:S - V41237:S	Numeric	Yes	Yes	Single (16-Bit) V-memory registers signed decimal ranging from -32768 to 32767
VC0 – VC3760	Numeric	Yes	Yes	Single (16-Bit) V-memory word registers decimal Aliases for mapped Control Relays C0 – C3777
VC0:B – VC3760:B	Numeric	Yes	Yes	Single (16-Bit) V-memory word registers BCD Aliases for mapped Control Relays C0 – C3777
VCT0 – VCT360	Numeric	Yes	Yes	Single (16-Bit) V-memory word registers decimal Aliases for mapped Counter Status (done) bits CT0 – CT377
VCT0:B – VCT360:B	Numeric	Yes	Yes	Single (16-Bit) V-memory word registers BCD Aliases for mapped Counter Status (done) bits CT0 – CT377
VGX0 – VGX3760	Numeric	Yes	Yes	Single (16-Bit) V-memory word registers decimal Aliases for mapped Remote I/O Inputs GX0 – GX3777
VGX0:B – VGX3760:B	Numeric	Yes	Yes	Single (16-Bit) V-memory word registers BCD Aliases for mapped Remote I/O Inputs GX0 – GX3777
VGY0 – VGY3760	Numeric	Yes	Yes	Single (16-Bit) V-memory word registers decimal Aliases for mapped Remote I/O Outputs GY0 – GY3777

Table 3-8. 105/205/350/405 DirectLogic PLC Data Members

VGY0:B – VGY3760:B	Numeric	Yes	Yes	Single (16-Bit) V-memory word registers BCD Aliases for mapped Remote I/O Outputs GY0 – GY3777
VS0 – VS1760	Numeric	Yes	Yes	Single (16-Bit) V-memory word registers decimal Aliases for mapped Stage status (active) bits S0 – S1777
VS0:B – VS1760:B	Numeric	Yes	Yes	Single (16-Bit) V-memory word registers BCD Aliases for mapped Stage status (active) bits S0 – S1777
VSP0 – VSP760	Numeric	Yes	Yes	Single (16-Bit) V-memory word registers decimal Aliases for mapped Special Relays (system status bits) SP0 – SP777
VSP0:B – VSP760:B	Numeric	Yes	Yes	Single (16-Bit) V-memory word registers BCD Aliases for mapped Special Relays (system status bits) SP0 – SP777
VT0 – VT360	Numeric	Yes	Yes	Single (16-Bit) V-memory word registers decimal Aliases for mapped Timer Status (done) bits T0 – T377
VCT0:B – VCT360:B	Numeric	Yes	Yes	Single (16-Bit) V-memory word registers BCD Aliases for mapped Timer Status (done) bits T0 – T377
VX0 – VX1760	Numeric	Yes	Yes	Single (16-Bit) V-memory word registers decimal Aliases for mapped Inputs X0 – X1777
VX0:B – VX1760:B	Numeric	Yes	Yes	Single (16-Bit) V-memory word registers BCD Aliases for mapped Inputs X0 – X1777
VY0 – VY1760	Numeric	Yes	Yes	Single (16-Bit) V-memory word registers decimal Aliases for mapped Outputs Y0 – Y1777
VY0:B – VY1760:B	Numeric	Yes	Yes	Single (16-Bit) V-memory word registers BCD Aliases for mapped Outputs Y0 – Y1777

Table 3-8. 105/205/350/405 DirectLogic PLC Data Members

X0 – X1777	Logical	Yes	Yes	Inputs – addressed in octal and mapped to V40400 – V40477
Y0 – Y1777	Logical	Yes	Yes	Outputs – addressed in octal and mapped to V40500 – V40577

305/305S Direct Logic PLC Family Data Members

Table 3-9. 305/305S DirectLogic Data Members

Data Member	Type	Read	Write	Description
Activated	Logical	Yes	No	Object-generated signal when TRUE, this flag signifies an active communication connection between the process file and the PLC.
C160 – C373 C1000 – C1067	Logical	Yes	No	Control Relays – addressed in octal
C374 – C377 C770 – C777 C1070 – C1077	Logical	Yes	No	Special Relays – addressed in octal
Failed	Logical	Yes	No	Object-generated signal when TRUE, this flag signifies the process file is no longer communicating with the PLC.
IO0 – IO157 IO700 – IO767	Logical	Yes	No	Inputs and Outputs – addressed in octal
Paused	Logical	Yes	No	Object-generated signal when TRUE, this flag signifies that the communication connection has paused but is still active and will go FALSE when its paused condition is satisfied. Usually caused by modifying the Link while it is in use.
R0 – R777	Numeric	Yes	Yes	Single (8-Bit) byte registers decimal

Table 3-9. 305/305S DirectLogic Data Members

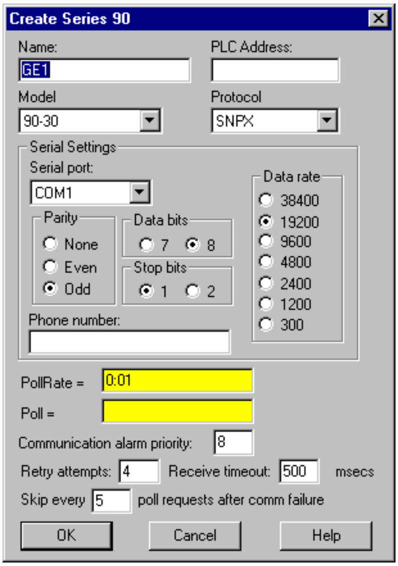
Data Member	Type	Read	Write	Description
R0:W – R776:W	Numeric	Yes	Yes	Single (16-Bit) word registers decimal
R0:WB – R776:WB	Numeric	Yes	Yes	Single (16-Bit) word registers BCD
RC160 – RC370 RC760RC1000 – RC1070	Numeric	Yes	Yes	Single (8-Bit) byte Control Relay and Special Relay registers decimal Aliases.
RC160:W – RC360:W RC760:W RC1000:W – RC1060:W	Numeric	Yes	Yes	Single (16-Bit) word Control Relay and Special Relay registers decimal Aliases.
RC160:WB – RC360:WB RC760:WB RC1000:WB – RC1060:WB	Numeric	Yes	Yes	Single (16-Bit) word Control Relay and Special Relay registers BCD Aliases.
RIO0 – RIO150 RIO700 – RIO760	Numeric	Yes	Yes	Single (8-Bit) byte registers decimal Aliases for mapped Inputs and Outputs IO0 – IO157 and IO700 – IO767
RIO0:W – RIO140:W RIO700:W – RIO750:W	Numeric	Yes	Yes	Single (16-Bit) word registers decimal Aliases for mapped Inputs and Outputs IO0 – IO157 and IO700 – IO767
RIO0:WB – RIO140:WB RIO700:WB – RIO750:WB	Numeric	Yes	Yes	Single (16-Bit) word registers BCD Aliases for mapped Inputs and Outputs IO0 – IO157 and IO700 – IO767
RS400 – RS570	Numeric	Yes	Yes	Single (8-Bit) byte Shift Registers decimal

Table 3-9. 305/305S DirectLogic Data Members

Data Member	**Type**	**Read**	**Write**	**Description**
RS400:W – RS560:W	Numeric	Yes	Yes	Single (16-Bit) word Shift Registers decimal
RS400:WB – RS560:WB	Numeric	Yes	Yes	Single (16-Bit) word Shift Registers BCD
SR400 – SR577	Logical	Yes	No	Shift Register Status Bits – addressed in octal
T600 – T677	Logical	Yes	No	Timer/Counter status (done) bits – addressed in octal
TCA600:WB – TCA677W:WB	Numeric	Yes	Yes	Timer/Counter current value words BCD – addressed in octal and mapped to R600 – R677

GE_Series90

GE_Series90 is a protocol driver class Lookout*Direct* uses to communicate with GE Series 90-30 and GE Series 90-70 programmable logic controllers (PLCs) using SNPX, a Series Ninety Protocol.


Create Series 90

Name: GE1

PLC Address:

Model: 90-30

Protocol: SNPX

Serial Settings

Serial port: COM1

Parity: ○ None ○ Even ● Odd

Data bits: ○ 7 ● 8

Stop bits: ● 1 ○ 2

Data rate: ○ 38400 ● 19200 ○ 9600 ○ 4800 ○ 2400 ○ 1200 ○ 300

Phone number:

PollRate = 0:01

Poll =

Communication alarm priority: 8

Retry attempts: 4 Receive timeout: 500 msecs

Skip every 5 poll requests after comm failure

PLC Address is a slave address and refers to the PLC address setting as configured on the device. The address can be up to eight ASCII characters.

Model chooses either 90-30 or 90-70.

Interface selects the protocol. You can choose between SNPX and Ethernet.

Serial port specifies which port the object uses for communication to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports…** command.

Data rate, **Parity**, **Data bits**, and **Stop bits** reference the settings on the hardware device.

Phone number specifies the number to be dialed if the serial port setting is configured for dial-up. This number only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. GE_Series90 then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *How LookoutDirect Works*, of the *Getting Started with LookoutDirect* manual for information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout*Direct* polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of object-generated alarms (0 – 10).

Retry attempts specifies the consecutive number of times Lookout*Direct* attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the GE_Series90 object generates an alarm and releases the communication port back to the communications service which then moves on to the next device in the polling queue (if any). Refer to Chapter 3, *Serial Communications*, in the *LookoutDirect Devleoper's Manual* for more information.

Receive timeout is the time delay Lookout*Direct* uses in waiting for a response from a device before retrying the request.

The **Skip every…** setting instructs Lookout*Direct* to not poll a device it has lost communication with on every scheduled poll. Instead, Lookout*Direct* skips the device in the polling cycle accordingly. Once communications have been reestablished, the device is polled on its regular cycle.

GE_Series90 Data Members

This protocol driver object contains a great deal of data. All readable and writable members (inputs/outputs), polling instructions, read/write blocking, serial port usage, and so on, are bundled with the object. Therefore, as soon as you create a GE_Series90 object you immediately have access to all the object data members (see data member list in Table 3-10).

Note Lookout*Direct* protocol driver objects automatically generate an efficient read/write blocking scheme based on the inputs and outputs being used in your process file. You are not required to build your own I/O blocking table.

Table 3-10. GE_Series90 Data Members

Data Member	Type	Read	Write	Description
AI1 – AI64	numeric	yes	yes	16-bit analog inputs encoded as unsigned binary integers ranging from 0 to 65535
AID1 – AID64	numeric	yes	yes	32-bit analog inputs encoded as unsigned binary integers ranging from 0 to 65535
AQ1 – AQ64	numeric	yes	yes	16-bit analog outputs encoded as unsigned binary integers ranging from 0 to 65535
AQD1 – AQD64	numeric	yes	yes	32-bit analog outputs encoded as unsigned binary integers ranging from 0 to 65535
CommFail	logical	yes	no	Object-generated signal that is on if, for whatever reason, Lookout*Direct* cannot communicate with the PLC.
I1 – I512	logical	yes	no	Single bit discrete inputs
M1 – M4096	logical	yes	yes	Single bit discrete (Internal coil)
OffHook	logical	no	yes	When TRUE, this flag instructs the GE object to retain exclusive use of its assigned communication port.
Poll	logical	no	yes	When this value transitions from FALSE to TRUE, Lookout*Direct* polls the device.
PollRate	numeric	no	yes	Lookout*Direct* expression that determines the device polling frequency.
Q1 – Q512	logical	yes	yes	Single bit discrete outputs
R1 – R9999	numeric	yes	yes	16-bit holding registers encoded as unsigned binary integers ranging from 0 to 65535

Table 3-10. GE_Series90 Data Members (Continued)

Data Member	Type	Read	Write	Description
RD1 – RD9998	numeric	yes	yes	32-bit holding registers encoded as unsigned binary integers ranging from 0 to 65535
S1 – S32	logical	yes	no	System fault
SA1 – SA32	logical	yes	no	Special Contacts A
SB1 – SB32	logical	yes	no	Special Contacts B
SC1 – SC32	logical	yes	no	Special Contacts C
Update	logical	yes	no	Object-generated signal that pulses each time the driver polls the device

GE_Series90 Status Messages

No response within timeout period

Lookout*Direct* did not received the expected response within the **Receive timeout** period. The object sent an inquiry and received an acknowledgment, but the device did not send an expected response to the request. This might happen if the response was interrupted. You may have to increase **Receive timeout**.

No return inquiry response from secondary unit

Lookout*Direct* received no response from the device within the **Receive timeout** period. The driver object is able to use the COM port, but when it polls the device, it does not respond—as if it is not even there. You may have to increase **Receive timeout** to ensure Lookout*Direct* is allowing enough time to receive the expected response. Also, verify your baud rate settings, cable connections, power, configuration settings, COM port settings, and polling addresses.

Bad LRC or BCC

The object is receiving a poll response from the device, but it could not decipher the response because it is garbled. Verify that all devices connected to the COM port have unique addresses. The last part of the message may actually be getting clipped off before it is completed. Consider increasing the number of **Retry attempts**. You may have to increase the **Receive gap** Serial Port setting to ensure Lookout*Direct* is receiving the entire message. If your Serial Port is configured for radio, this could be caused by an audible squelch tail occurring at the end of a radio transmission. Try adjusting **RTS delay off** and **CTS timeout**.

No attach response within timeout period

An attempt was made to establish communications with the PLC without any response. Check your cabling and COM port selections, power, configuration settings, and polling addresses.

Invalid response [x]

An error in the structure of a response frame was detected. You may have two PLCs with the same address.

Incorrect response length [x]

A response was received with an unexpected length. You may have to increase the **Receive gap** Serial Port setting to ensure Lookout*Direct* is receiving the entire message.

Incorrect response Address

A response was received with an address not matching the objects address. You may have two master devices on the network.

SNPX ERROR—Major code: x Minor code: x

The response message contained an SNPX error code. Refer to your GE documentation for the meaning of this particular error.

Modbus

Modbus and ModbusMOSCAD are protocol driver classes Lookout*Direct* uses to communicate with equipment such as programmable logic controllers (PLCs), remote terminal units (RTUs), or any other piece of equipment using Modbus Serial (ASCII or RTU) or Modbus Plus communication protocol.

The Modbus object class has general-purpose addresses, such as holding register 40001, and is suitable for communicating with nearly all Modbus devices, including the Control Microsystems TeleSAFE RTU.

The ModbusMOSCAD object class works with Motorola MOSCAD PLCs and RTUs. It also uses the Modbus Serial (ASCII or RTU) or Modbus Plus communication protocol, but its data members reflect the address of Motorola MOSCAD devices.

You can limit the number of channels Lookout*Direct* uses on the SA-85 card by creating a `modbus.ini` file in the Lookout*Direct* directory, as shown in the following example.

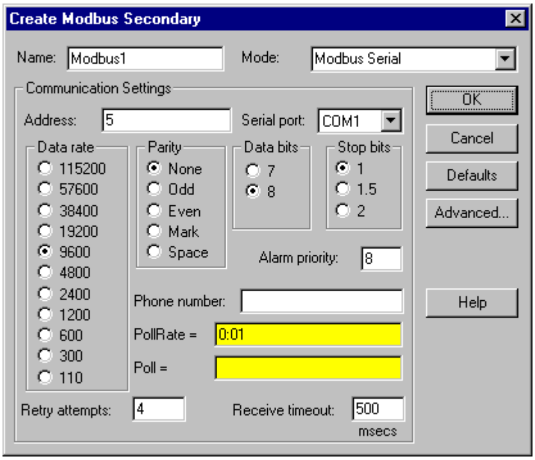
```
[ALL]
MaxChannels=channel
```

where *`channel`* is a number between 1 and 8, inclusive (default = 8).

These protocol driver objects contain a great deal of data. All readable and writable members (inputs/outputs), polling instructions, read/write blocking, serial port usage, and so on are bundled with the object. Therefore, as soon as you create a Modbus or ModbusMOSCAD object you immediately have access to all the object data members (see data member list in Table 3-12).

Note Lookout*Direct* protocol driver objects automatically generate an efficient read/write blocking scheme based on the inputs and outputs being used in your process file. You are not required to build your own I/O blocking table.

As protocol drivers, both object classes conform to the specifications in the Modicon Modbus Protocol Reference Guide PI-MBUS-300 Rev. C. The drivers support ASCII and RTU transmission modes, as well as Modbus Plus.



Name: Modbus1 Mode: Modbus Serial

Communication Settings

Address: 5 Serial port: COM1

Data rate: ○ 115200 ○ 57600 ○ 38400 ○ 19200 ● 9600 ○ 4800 ○ 2400 ○ 1200 ○ 600 ○ 300 ○ 110

Parity: ● None ○ Odd ○ Even ○ Mark ○ Space

Data bits: ○ 7 ● 8

Stop bits: ● 1 ○ 1.5 ○ 2

Alarm priority: 8

Phone number:

PollRate = 0:01

Poll =

Retry attempts: 4 Receive timeout: 500 msecs

In this example, Lookout*Direct* is connected to a Modbus-speaking PLC with an address of 5 using serial port 1 (which was previously configured for hardwired communications), and polling the device every second.

Modbus Serial indicates that the slave device talks either Modbus ASCII or Modbus RTU. When you select this option, Lookout*Direct* first tries to communicate using the RTU format. If unsuccessful, it then tries the ASCII format (a little slower). If your network is susceptible to repeated communication problems, and if these problems slow scanning considerably, you may want to disable Lookout*Direct* from retrying both formats. This can speed communication retries by Lookout*Direct*; however, it will not fix your communication problems. Call National Instruments technical support to for information on how to prohibit Lookout*Direct* from trying to communicate using both formats.

Modbus Plus Network indicates that the slave device is connected to the Lookout*Direct* computer via a Modbus Plus network card.

Note NetBIOS-based networking software typically uses software interrupt 5C. This is also the default software interrupt used by the Modbus Plus Network card driver. Change the Modbus Plus software interrupt from 5C to 5D, 5E, or 5F. Refer to your Modicon documentation for instructions on changing the software interrupt setting.

Modbus Ethernet indicates you are communicating with the Modbus slave device using an Ethernet connection.

If you select **Modbus Serial**, you must specify **Address**, **Serial Port**, **Data Rate**, **Parity**, **Data Bits**, and **Stop Bits**. And if you are using a Dial-up modem connected to your communication port, you must also specify a **Phone Number**.

If you select **Modbus Plus Network**, you need only specify the remote device **Address**.

If you select **Modbus Ethernet** you must specify the **IP address** in addition to **Alarm Priority**, **Poll Rate**, **Poll Retry attempts**, and **Receive timeout**.

Address is a slave address and refers to the PLC or RTU address setting as set on the device dip switches. If devices share a common line, they require unique addresses (1 to 255).

IP address is the Internet Protocol address for the Modbus slave object you are communicating with.

Serial port specifies which port the object uses for communication to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports…** command.

Data rate, **Parity**, **Data bits**, and **Stop bits** reference the settings on the hardware device.

The **Defaults** button replaces the current settings with default values.

Alarm priority determines the priority level of Modbus-generated alarms.

Phone number specifies the number to be dialed if the serial port setting is configured for dial-up. This number only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. Modbus then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *How LookoutDirect Works*, of the *Getting Started with LookoutDirect* manual for information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout*Direct* polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

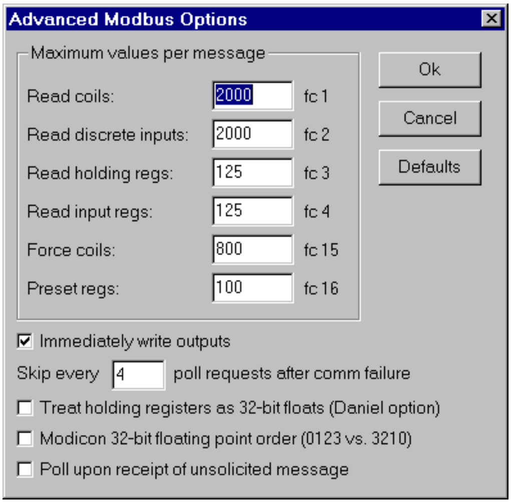
Retry attempts specifies the consecutive number of times Lookout*Direct* attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the Modbus object generates an alarm and releases the communication port back to the communications subsystem which then moves on to the next device in the polling queue (if any). Refer to Chapter 3, *Serial Communications*, in the *LookoutDirect Developer's Manual* for more information.

Receive timeout is the time delay Lookout*Direct* uses in waiting for a response from a device before retrying the request.

Advanced Modbus Parameters

The Modbus driver attempts to block the reads and writes of coils, input registers and holding registers into groups to maximize communication efficiency. Through the **Advanced Modbus Options** dialog box, you can control the maximum block sizes that the driver uses. In fact, if your device does not support the default block sizes, you may have to specify smaller blocks.

The **Advanced…** button invokes the **Advanced Modbus Options** dialog box you can use to customize specific options within the Modbus protocol.



Advanced Modbus Options

Maximum values per message

Read coils:	2000	fc 1
Read discrete inputs:	2000	fc 2
Read holding regs:	125	fc 3
Read input regs:	125	fc 4
Force coils:	800	fc 15
Preset regs:	100	fc 16

☑ Immediately write outputs

Skip every 4 poll requests after comm failure

□ Treat holding registers as 32-bit floats (Daniel option)

□ Modicon 32-bit floating point order (0123 vs. 3210)

□ Poll upon receipt of unsolicited message

The Modbus object class uses Modbus Function Codes 01, 02, 03, 04, 05, 06, 15, and 16; and expects the remote I/O device to support these codes as specified by Modbus. The driver can communicate with up to 247 Modbus slave devices on each serial port.

The **Maximum values per message** settings specify the maximum number of elements Lookout*Direct* attempts to read (`fc 1 – fc 4`), or write (`fc 15` and `fc 16`), in a single Modbus message. The default values represent the maximum number of elements that the protocol can transmit in a single message, and provides optimal speed. However, some devices are not capable of handling the maximum number of elements, so you should set the values according to the documentation for those devices.

If the **Immediately write outputs** option is ON, Lookout*Direct* immediately polls the device any time a value changes that is being written out to the device. If it is OFF, Lookout*Direct* waits until the next scheduled poll to write out changed values.

The **Skip every…** setting instructs Lookout*Direct* to not poll a device it has lost communication with on every scheduled poll. Instead, Lookout*Direct* skips the device in the polling cycle accordingly. Once communications have been reestablished, the device is polled on its regular cycle.

The **Daniel option** is device-dependent and instructs Lookout*Direct* to treat holding registers as 32-bit IEEE floating values instead of 16-bit values. If you set this flag, you must also set your hardware device to treat holding registers as 32-bit floats—most devices do not support this option, but Bristol-Babcock RTUs and Daniel flow meters do.

Note Activating the Daniel option deactivates all Modbus holding register members (on that device) except for 40001 – 49999 and 4000001 – 465000. If you attempt to read D40001, for example, the returned value is 0, and Lookout*Direct* will not attempt to write D40001 to the RTU. Of course, in devices that do not support this option, you can still read and write two adjacent holding registers as a floating point value with the Modbus data members F40001 – F4999. In fact, this is a more general purpose solution than the Daniel option, because you can still read bits and word values out of the holding registers, too.

Some Modbus PLCs reverse byte order in a floating point data member. If your process is returning garbled or senseless floating point values, select the **Modicon 32-bit floating point order (0123 vs. 3210)** option in the **Advanced Modbus Options** dialog box. This option chooses whether the characters within the floating point registers (data members `F40001 – F49999` and `F400001 – F465000`) are in little endian or big endian format.

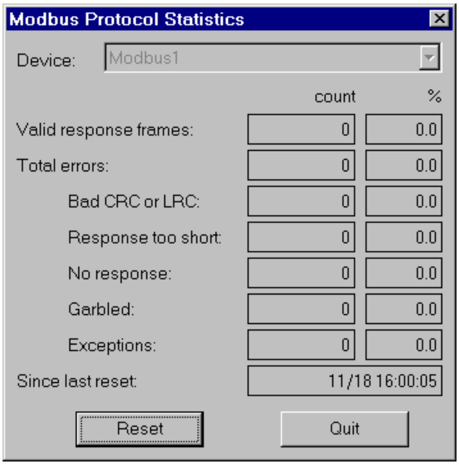
On occasion a Modbus PLC will *direct* a message to your process. To poll for all data member values following receipt of such a poll, select the **Poll on receipt of unsolicited message** option in the **Advanced Modbus Options** dialog box. Using this option is a way to force a poll of a Modbus device based on an event, without reference to the configured poll intervals and somewhat outside the usual Lookout*Direct* event-driven action.

Caution Notice that the D and F data members read two adjacent registers as single, 32-bit numbers. One consequence of this is that if you connect to two adjacent registers, such as **`D40010`** and **`D40011`**, you will get incorrect values because of overlapping. Make sure that you do not connect to adjacent registers with the D or F data members.

Modbus Protocol Statistics

The driver monitors Modbus Protocol Statistics. This data is held within readable data members of the Modbus object and you can see them in the Modbus Protocol Statistics dialog box. To view the dialog box, select **Options»Modbus…** and click on **Statistics…**.

Note The **Options»Modbus…** option is only visible in the **Options** menu if a Modbus object was previously created in your Lookout*Direct* application.



Device: Modbus1

	count	%
Valid response frames:	0	0.0
Total errors:	0	0.0
Bad CRC or LRC:	0	0.0
Response too short:	0	0.0
No response:	0	0.0
Garbled:	0	0.0
Exceptions:	0	0.0
Since last reset:	11/18 16:00:05	

The **Count** column contains the accumulated number of messages received from the selected **Device** that fall into each respective category since the last time the Reset button was pressed. The percent column (**%**) indicates the percentage of messages received that fall into each respective category since the last time the Reset button was pressed.

When you depress the **Reset** button, the **ResetCounts** data member is set TRUE, setting all statistical values to zero. Lookout*Direct* records the date and time that the reset was last performed in the **Since last reset** data field.

Modbus Data Members

The Modbus object class supports both 5-digit and 6-digit addressing. When you use a 6 digit address, the left-most digit represents the address *type* as follows:

Table 3-11. 6-Digit Address Coding

First Digit	Address Type
0	Single-bit coils
1	Discrete inputs
3	Input registers
4	Holding registers

The remaining 5 digits represent the actual address of the coil, input or holding register.

Note When you reference address 000001 and address 1, you are referring to the same point, but 40001 and 040001 do not refer to the same point. Because zero is the left-most digit in the 6-digit address, 040001 points to the 40001st single-bit coil and 40001 refers to the first holding register.

Table 3-12. Modbus Data Members

Data Member	Type	Read	Write	Description
000001 – 065000	logical	yes	yes	6-digit addresses of single-bit coils
1 – 9999	logical	yes	yes	Single-bit coils
100001 – 165000	logical	yes	no	6-digit addresses of single-bit discrete inputs
10001 – 19999	logical	yes	no	Single bit discrete inputs
300001 – 365000	numeric	yes	no	6-digit addresses of 16-bit input registers encoded as unsigned binary integers ranging from 0 to 65535
30001 – 39999	numeric	yes	no	16-bit input registers encoded as unsigned binary integers ranging from 0 to 65535

Table 3-12. Modbus Data Members (Continued)

Data Member	Type	Read	Write	Description
400001 – 465000	numeric	yes	yes	6-digit addresses of 16-bit input registers encoded as unsigned binary integers ranging from 0 to 65535
400001.1 – 465000.16	logical	yes	yes	6-digit address used to access individual bits out of holding registers and read them as logical ON/OFF values. The least significant bit is 1; the most significant, 16.
40001 – 49999	numeric	yes	yes	16-bit holding registers encoded as unsigned binary integers ranging from 0 to 65535
40001.1 – 49999.16	logical	yes	yes	Access individual bits out of holding registers and read them as logical ON/OFF values. The least significant bit is 1; the most significant, 16.
BadCRC	numeric	yes	no	Number of responses from device whose message failed the cyclic redundancy check (CRC) or the longitudinal redundancy check (LRC)
BCD300001 – BCD365000	numeric	yes	no	6-digit addresses of 16-bit input registers encoded as binary-coded decimal integers ranging from 0 to 9999
BCD30001 – BCD39999	numeric	yes	no	16-bit input registers encoded as binary-coded decimal integers ranging from 0 to 9999
BCD400001 – BCD465000	numeric	yes	yes	6-digit addresses of 16-bit holding registers encoded as binary-coded decimal integers ranging from 0 to 9999
BCD40001 – BCD49999	numeric	yes	yes	16-bit holding registers encoded as binary-coded decimal integers ranging from 0 to 9999

Table 3-12. Modbus Data Members (Continued)

Data Member	Type	Read	Write	Description
CommFail	logical	yes	no	Driver-generated signal that is ON if Lookout*Direct* cannot communicate with the device for whatever reason
D400001 – D465000	numeric	yes	yes	6-digit addresses of 32-bit unsigned holding register—reads two adjacent holding registers as a single 32-bit number ranging from 0 to 4,294,967,296.
D40001 – D49999	numeric	yes	yes	32-bit unsigned holding register—reads two adjacent holding registers as a single 32-bit number ranging from 0 to 4,294,967,296.
Exceptions	numeric	yes	no	Number of responses from device whose message was understandable to the driver but included an error code indication from the device
F400001 – F465000	numeric	yes	yes	6-digit addresses of 32-bit IEEE floating point register—reads two adjacent holding registers as a single 32-bit floating point value
F40001 – F49999	numeric	yes	yes	32-bit IEEE floating point register—reads two adjacent holding registers as a single 32-bit floating point value
Garbled	numeric	yes	no	Number of responses from device whose message was unintelligible to the driver
NoResponse	numeric	yes	no	Number of polls generated by driver not responded to by device
OffHook	logical	no	yes	When TRUE, this flag instructs the Modbus object to retain exclusive use of its assigned communication port
Poll	logical	no	yes	When this expression transitions from FALSE to TRUE, Lookout*Direct* polls the device.

Table 3-12. Modbus Data Members (Continued)

Data Member	Type	Read	Write	Description
PollRate	numeric	no	yes	Lookout*Direct* expression that determines the device polling frequency.
ProtocolErrors	numeric	yes	no	Total number of bad messages received from polled device
ResetCounts	logical	no	yes	Resets number to zero in the following data members: ValidFrame, NoResponse, TooShort, BadCRC, Garbled, Exceptions, & ProtocolErrors
S400001 – S465000	numeric	yes	yes	6-digit addresses of 16-bit holding registers encoded as signed binary integers ranging from –32767 to +32768.
S40001 – S49999	numeric	yes	yes	16-bit holding registers encoded as signed binary integers ranging from –32767 to +32768.
TooShort	numeric	yes	no	Number of responses from device whose message length was too short
Update	logical	yes	no	Driver-generated signal that pulses each time the driver polls the device
ValidFrame	numeric	yes	no	Number of good messages received from polled device

Comments You can use the OffHook data member to enhance communications when using the Modbus object class with dial-up modems. When OffHook is TRUE and the serial port is connected to a dial-up modem, the Modbus object does not hang up the modem when the poll is complete. Rather, it keeps the phone off the hook, retaining exclusive use of the serial port. As long as OffHook is TRUE, the Modbus object continues to poll the same PLC without hanging up the modem.

As soon as OffHook goes FALSE, the object releases the serial port to the communications subsystem, which goes to the next poll request in the queue, if any. The object also releases the port if data communications are lost for any reason—such as if the PLC modem breaks the connection.

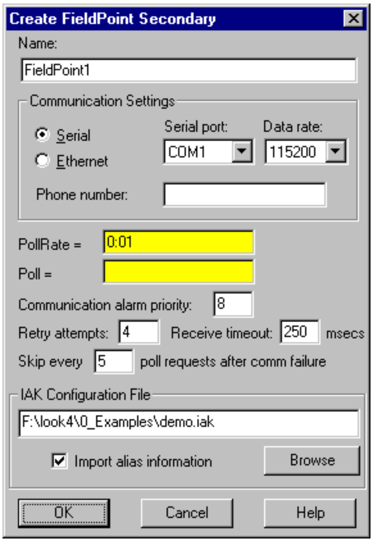
When using OffHook, consider defining the driver object **PollRate** to poll fast when OffHook is TRUE, and poll at its normal rate when OffHook is FALSE. You might tie a Switch object to the OffHook writable data member for this very purpose.

National Instruments FieldPoint

FieldPoint is a protocol driver class Lookout*Direct* uses to communicate with FieldPoint devices using an enhanced version of the Optomux communication protocol. This object works with the FieldPoint models FP-1000, FP-1001, FP-AI-110, FP-AO-200, FP-DI-330, and FP-DO-400.

This protocol uses no parity, eight data bits and one stop bit. In Lookout*Direct*, a single FieldPoint object represents all devices connected to the same COM port.

The Lookout*Direct* FieldPoint object can read and write to all predefined data points allowed by the particular FieldPoint module. When you create a FieldPoint object, you have immediate access to all the object data members. See the *FieldPoint Data Members* section for more information on object data members.



Create FieldPoint Secondary

Name: FieldPoint1

Communication Settings

● Serial ○ Ethernet

Serial port: COM1

Data rate: 115200

Phone number:

PollRate = 0:01

Poll =

Communication alarm priority: 8

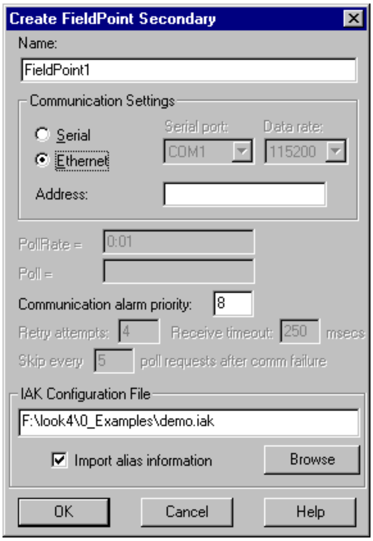
Retry attempts: 4 Receive timeout: 250 msecs

Skip every 5 poll requests after comm failure

IAK Configuration File

F:\look4\0_Examples\demo.iak

☑ Import alias information



Create FieldPoint Secondary

Name: FieldPoint1

Communication Settings

○ Serial ● Ethernet

Serial port: COM1

Data rate: 115200

Address:

PollRate = 0:01

Poll =

Communication alarm priority: 8

Retry attempts: 4 Receive timeout: 250 msecs

Skip every 5 poll requests after comm failure

IAK Configuration File

F:\look4\0_Examples\demo.iak

☑ Import alias information

When you select the **Ethernet** option, the serial port configuration options are disabled.

Enter the IP address in the **Address** field. You could also enter the FieldPoint network name instead.

Note If you replace a FieldPoint FP-1000 module (serial communications) with a FieldPoint FP-1600 module (Ethernet communications) or replace an FP-1600 module with an FP-1000 module, you must re-import your `.IAK` file. First, you must use

FieldPoint Explorer to update your `.IAK` file. If necessary, consult your FieldPoint online help for detailed instructions on using FieldPoint Explorer. To re-import the `.IAK` file, unselect the **Import alias information** checkbox and click on **OK**. Reopen the FieldPoint object dialog box and select the **Import alias information** checkbox again. When you click on **OK**, Lookout*Direct* re-imports the new `.IAK` file infomation.

Serial port specifies which COM port the object uses for communicating to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports...** menu command.

Data rate indicates the baud rate that Lookout*Direct* uses to communicate with the hardware device. This setting should match the selection made on each of your network modules.

Phone number specifies the number to be dialed if the selected serial port is configured for dial-up. This number only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. The object then polls the device at the specified time interval. Ordinarily, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *How LookoutDirect Works*, of the *Getting Started with LookoutDirect* manual for information on entering time constants.

Poll is a logical expression. When transitioned from FALSE to TRUE, Lookout*Direct* polls the device. This can be a simple expression, like the signal from a pushbutton, or it can be a complex algorithm.

Communication alarm priority determines the priority level of alarms generated by the FieldPoint object. Such alarms are typically related to communications with the physical device.

Retry attempts specifies the consecutive number of times Lookout*Direct* attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the FieldPoint object generates an alarm and releases the COM port. Refer to Chapter 3, *Serial Communications*, in the *LookoutDirect Developer's Manual* for more information.

Receive timeout is the time delay Lookout*Direct* uses in waiting for a response from a device before retrying the request.

The **Skip every n** setting instructs Lookout*Direct* to not poll a device it has lost communication with on every scheduled poll. Instead, Lookout*Direct* skips the device in the polling cycle. Once communications have been reestablished, the device is polled on its regular cycle.

IAK configuration file is a dialog for selecting an IAK configuration file. The IAK file contains alias and scaling information which is extracted for use in Lookout*Direct*. Choose the configuration file you want to use by entering the path directly, or use the **Browse** button. Check **Import alias information** if you want to extract information from the selected file.

FieldPoint Data Members

As with all Lookout*Direct* drivers, you can access I/O points and other data through data members. The following is a table of data members currently supported by the FieldPoint object class.

Table 3-13. National Instruments FieldPoint Data Members

Data Member	Type	Read	Write	Description
AI000.00 - AI255.15	numeric	yes	no	Analog input channels.
AO000.00 - AO255.15	numeric	yes	yes	Analog output channels.
CA000.00 - CA255.15	numeric	yes	no	Use to connect to count inputs on a FieldPoint counter module.
CD000.00 - CD255.15	logical	yes	no	Use to connect to digital inputs on a FieldPoint counter module.
CI000.00 - CI255.15	logical	no	yes	Counter increment.
CO000.00 - CO255.15	logical	yes	yes	Use to connect to digital outputs in a FieldPoint Counter module.
CommFail	logical	yes	no	Goes high if Lookout*Direct* cannot communicate with the device.
CR000.00 - CR255.15	logical	no	yes	Counter reset.
DI000.00 - DI255.15	logical	yes	no	Discrete input channels.
DO000.00 - DO255.15	logical	yes	yes	Discrete output channels.
PF000.00 - PF255.15	numeric	yes	yes	Pulse width modulator period setting. Period is the entire on/off time of the pulse in milliseconds.

Table 3-13. National Instruments FieldPoint Data Members (Continued)

Data Member	Type	Read	Write	Description
Poll	logical	no	yes	When transitioned from low to high, Lookout*Direct* begins a poll cycle on the device.
PollRate	numeric	no	yes	Specifies the frequency at which Lookout*Direct* polls the device.
PW000.00 - PW255.15	numeric	yes	yes	Sets the FieldPoint pulse module duty cycle on/off ratio. A setting of 10.00 means the pulse is on for 10% of the pulse period and off for 90%.
RL000.00 - RL255.15	logical	yes	yes	Use to connect to digital output channels on a FieldPoint relay module.
RT000.00 - RT255.15	numeric	yes	no	RTD temperature measurement module analog input; returns a value in degrees Centigrade.
TC000.00 - TC255.15	numeric	yes	no	Use to connect to analog input channels on a FieldPoint thermocouple module.
Update	logical	yes	no	Goes high when Lookout*Direct* begins a poll cycle on the device.

Note When you use the `CA`, `CD`, `CI`, `CO`, `CR`, `PW`, `RL`, `RT`, and `TC` data members with FieldPoint serial modules, they are synonyms for analog or digital inputs or outputs as follows:
CI, CD—Digital Input
CR, CA, RL—Digital Output
PF, PP, RT, TC—Analog Input
PW—Analog Output

With FieldPoint Ethernet, these data members represent different FieldPoint modules.

Note The first two characters of the I/O data members represent the kind of module being accessed. The next three digits represent the device address of the module. This is the address of the I/O module itself, not the network module that governs it. Following the period are two digits representing the channel number within the module.

Not all of these data members are valid for every FieldPoint module. For all the device types you are able to select the full range of device addresses and channels. So if you select DO123.03, you need to be certain that the device at address 123 is in fact a discrete output module.

For a more complete definition of the function of these data members, see FieldPoint documentation.

Note In the event of a power cycle to the FieldPoint device during use, the configuration of the device reverts to some default state, which is configurable. You should keep in mind that if the ranges you configured into the IAK file differ from those in the power-up configuration, the scaling information imported from the IAK file and used as a Lookout*Direct* alias might become outdated and incorrect after a power loss. To avoid this, make certain your power-up configuration ranges and your IAK configuration ranges are identical.

FieldPoint Multiple Discrete Data Members

Table 3-14. Multiple Discrete Data Members

Data Member	Type	Read	Write	Description
MDI000.0000 - MDI255.FFFF	numeric	yes	no	Multiple discrete input channels
MDO000.0000 - MDO255.FFFF	numeric	yes	yes	Multiple discrete output channels

These special purpose data members are for reading or writing a numeric integer value to a set of discrete channels.

For instance, when you are configuring your modules with FieldPoint Explorer, you have the option of selecting more than one discrete channel for a data item that you are defining. If you do this and import the resulting `.IAK` file into Lookout*Direct* for use as aliases, the aliases created will correspond to this set of data members. You can then read and write to all the discrete channels with a single numeric data member. The data member names are in the form `MTTAAA.CCCC`, where:

`M`Indicates multiple as opposed to single

`TT`Two characters specifying module type (Discrete Out, Discrete In)

`AAA`Three numeric characters specifying module address

`CCCC`Four hexadecimal characters specifying which of the 16 channels are included in this data member

Note These data members will not enumerate. You may use them either by importing configurations from FieldPoint Explorer or by entering the data member name explicitly.

FieldPoint Error Messages

No response within timeout period

Lookout*Direct* received no response from a device within the **Receive timeout** period. The FieldPoint object is able to use the COM port, but when it polls the device, the device does not respond. If you have daisy-chained several devices, you have introduced an inherent delay. You may have to significantly increase **Receive timeout** (and **Poll Rate**) to ensure Lookout*Direct* is allowing enough time to receive the expected response. This increase has nothing to do with the processing capabilities of Lookout*Direct*. Rather it is based solely on **Data rate** and the number of devices on the chain. Also, verify your baud rate settings, cable connections, power, configuration settings, COM port settings, and polling addresses.

Module returning ?? checksum

This means that the frame sent from the PLC in response to the command sent by Lookout*Direct* out returned `??` instead of a valid checksum. Check FieldPoint configuration.

Message Garbled - Bad CRC

This means the checksum (`CRC` in this case) failed in a frame received by Lookout*Direct*. Check cabling or for two or more devices with the same address.

Unexpected data response length

The frame received was of an unexpected length. Check the Lookout*Direct* **receive gap** setting.

Error loading IAK configuration file

Lookout*Direct* was not able to successfully extract data from the `.IAK` configuration file. Try running the FieldPoint Explorer again and reconfigure your hardware.

FP error: Power-up clear expected

A command other than power-up clear was attempted after power-up or power failure. The command sent is ignored and normal operations should resume.

FP error: Undefined command

The addressed module does not support this command. (for example, trying to write to an input module) Check to see if you are sending a command appropriate to the module.

FP error: Checksum error

This means the checksum (`CRC` in this case) failed in a frame sent by Lookout*Direct*. Check the Lookout*Direct* **receive gap** setting.

FP error: Input buffer overrun

The command sent to the FieldPoint module was too long. Check the Lookout*Direct* **receive gap** setting.

FP error: Non-printable ASCII character received

Only characters from ASCII value 33 to 127 are permitted in FieldPoint commands. The command is ignored.

FP error: Data field error

An insufficient or incorrect number or characters were received by the FieldPoint module for the specified command. Check the Lookout*Direct* **receive gap** setting.

FP error: Communications link network watchdog timed out

There has been no network traffic in the amount of time specified by your watchdog configuration settings, and the system has reverted to its watchdog defaults.

FP error: Specified limits invalid for the command

This includes the case where an invalid digit (hex or decimal) was received. Check the Lookout*Direct* **receive gap** setting.

FP error: ASCII to binary conversion error

One or more ASCII characters could not be converted to binary on the FieldPoint module. Check the Lookout*Direct* **receive gap** setting.

FP error: Invalid device address

The command is valid, but the addressed module does not support the command received. Check to see if you are sending a command appropriate to the module.

FP error: Serial framing error

An improperly framed command was received by the FieldPoint module. Check the Lookout*Direct* **receive gap** setting.

FP error: Addressed module does not exist

Make sure that you are addressing a valid module address.

FP error: Invalid channel

One or more channels specified in the command either do not exist or do not support the operation specified.

FP error: Invalid range setting

Check to see that the range information on the module has not changed, possibly due to a loss of power.

FP error: Invalid operation for the module

One or more module-specific operations specified in the command either do not exist or do not support the operation specified. Make sure that you are not requesting a discrete operation for an analog module, and vice versa.

FP error: Module has been hotswapped since last command

The alarm should deactivate immediately after it appears. It appearance is only to acknowledge that a hot swap has occurred.

FP error: Irrecoverable hardware fault

A malfunction in the FieldPoint firmware or hardware has made communications from Lookout*Direct* impossible.

Channel specific error: dev:##,ch:##,err:##

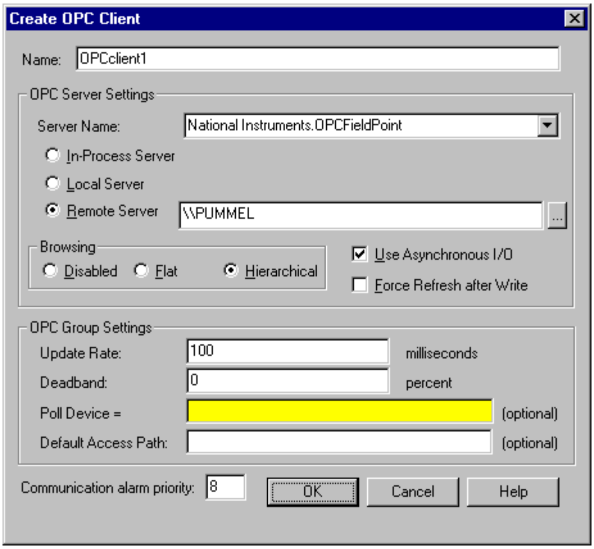
These are error codes returned from the FieldPoint I/O modules. The alarm message specifies a device address, channel number, and error code. See FieldPoint documentation for a description of the error condition.

National Instruments Lookout*Direct* OPC Client

Lookout*Direct* uses the National Instruments Lookout*Direct* OPC client to read data from and write data to any OPC server. It supports numeric, logical (boolean), and text I/O. The Lookout*Direct* 4 OPC client normally reads from the cache. You can connect a switch or pushbutton to the `PollDevice` data member to trigger a device update.

For your convenience, you can create OPCFieldPoint and OPCNIDAQ versions of the OPCClient object class by selecting these two special cases from the **Select Object Class** dialog box.

Note If you are using the National Instruments OPC IATest server, there is no device to poll. Triggering the PollDevice data member returns a zero (0). Simulated data will be restored shortly after such a poll.



Name: OPCclient1

OPC Server Settings

Server Name: National Instruments.OPCFieldPoint

○ In-Process Server

○ Local Server

● Remote Server \\PUMMEL

Browsing: ○ Disabled ○ Flat ● Hierarchical

☑ Use Asynchronous I/O

□ Force Refresh after Write

OPC Group Settings

Update Rate: 100 milliseconds

Deadband: 0 percent

Poll Device = (optional)

Default Access Path: (optional)

Communication alarm priority: 8

The **Server Name** box enumerates all of the OPC servers registered on the local computer. Select the appropriate server.

Note The Server Name listbox contains only local servers. It has no effect on what you find when you browse remote servers.

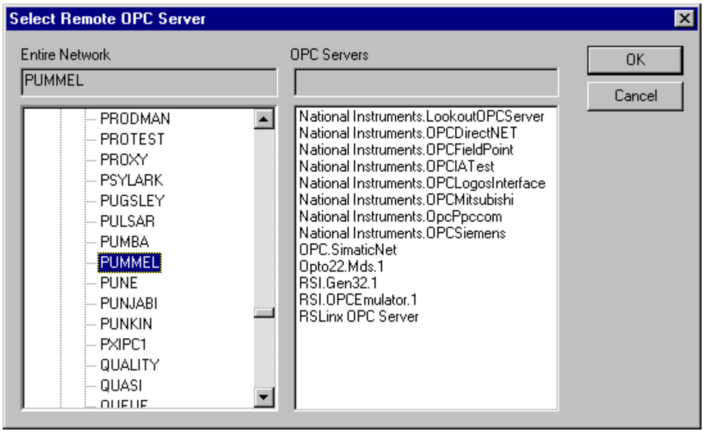
You can then select one of the **In-Process Server**, **Local Server**, and **Remote Server** options. These options specify the type of server the OPC client will attempt to launch. If it cannot successfully connect to the selected server, the OPC client generates an alarm.

Select **Remote Server** to enable **Computer Name**, which specifies the name of the computer on which the remote server is to be launched. If you know the computer name you want, enter it preceded with two backslashes, as in `\\PUMMEL`.

Caution In-process servers should only be used if Lookout*Direct* is the only program communicating with the server. If other programs address the same server, use the Local Server option.

The **Browsing** options include **Disabled**, **Flat**, and **Hierarchical**. If your OPC servers permit browsing, select either **Flat** or **Hierarchical**. If you enable flat browsing all data members appear in any Lookout*Direct* windows displaying OPC client data members. If you select **Hierarchical**, the data members are arranged in hierarchical folders.

To browse remote servers, click on the browsing button next to the **Remote Server** field. The following dialog box appears.


Select Remote OPC Server
OPC Servers
OK
Cancel
PRODMAN
PROTEST
PROXY
PSYLARK
PUGSLEY
PULSAR
PUMBA
PUMMEL
PUNE
PUNJABI
PUNKIN
PXIPC1
QUALITY
QUASI
QUEUE
National Instruments.LookoutOPCServer
National Instruments.OPCDirectNET
National Instruments.OPCFieldPoint
National Instruments.OPCIATest
National Instruments.OPCLogosInterface
National Instruments.OPCMitsubishi
National Instruments.OpcPpccom
National Instruments.OPCSiemens
OPC.SimaticNet
Opto22.Mds.1
RSI.Gen32.1
RSI.OPCEmulator.1
RSLinx OPC Server

Entire Network: PUMMEL

Browse for the computer you want, and select the OPC server running on the computer that you want to access.

Note You must have the Microsoft Remote Registry service installed for remote OPC browsing to work on a computer running Windows 98/95. To install, select **Start»Settings»Control Panel** and then select **Network**. Check the Configuration tab to see if Microsoft Remote Registry is installed. If it is not, click on the **Add** button, and select **Service** in the **Select Network Component Type** dialog box. Choose Microsoft as the manufacturer. If the Remote Registry service is not visible, you will need your Windows 98/95 CD-ROM. You may have to browse through the disk to find the correct service. Once you have installed this service, you can successfully use remote OPC browsing in the Lookout*Direct* OPCClient object.

Select **Use Asynchronous I/O** if you want to us asynchronous communications with your OPC server. This is the preferred communications mode, and should be used when possible.

The **Force Refresh after Write** option is only available if you select the **Use Asynchronous I/O** option. This option forces the OPC server to return the current status of all data members every time you write to one.

Note Selecting **Force Refresh after Write** can sometimes impair performance of your Lookout*Direct* process, depending on the number of data members in your OPC server and other system variables, including communications speed and what other tasks are being performed at any particular moment.

Update Rate is a numeric expression that determines how often the OPC server updates the client. Enter this time interval in milliseconds.

Deadband sets the percentage change that must take place in a data member before the OPC server reports a change in value to Lookout*Direct*.

Poll Device is a logical expression. When this expression changes from FALSE to TRUE, Lookout*Direct* polls the OPC server for all values. You can use a simple expression like the signal from a pushbutton, or a complex algorithm. In most applications there is no need to do device reads, so this parameter is often left blank.

Enter a **Default Access Path** if you want to simplify entering paths to various data members.

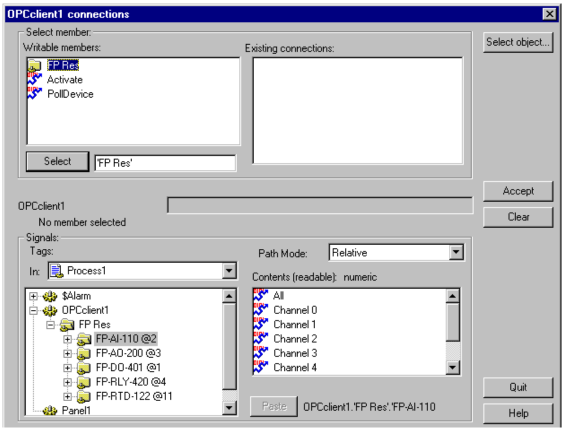
Communication Alarm Priority specifies the priority of alarms generated by this OPC client object.

OPC Client Data Members

As with all Lookout*Direct* drivers, you can access I/O points and other data through data members.

Unlike other Lookout*Direct* driver objects, however, the OPCClient data member set changes depending on the OPC servers you have running on your computer.

If you selected hierarchical browsing when you created the OPCClient, the data members for the OPC server you have connected your OPC client to appear as data members inside a folder in your OPCClient list of data members in the Lookout*Direct* Object Explorer, connection editor, and other windows as shown in the following illustration.



'FP Res'

OPCclient1
No member selected

In: Process1

Path Mode: Relative

Contents (readable): numeric

OPCclient1.'FP Res'.'FP-AI-110

If you selected flat browsing, you will see a long list of available data members without hierarchical organization into folders.

If you did not enable browsing, or your computer cannot browse certain OPC servers, you will only see the data members built into the Lookout*Direct* OPCClient.

The Lookout*Direct* OPCClient object class currently contains the built-in data members contained in the following table.

Table 3-15. OPCClient Data Members

Data Member	Type	Read	Write	Description
Activate	logical	no	yes	When TRUE, the OPC client is active and receives data. When FALSE, the OPC client does not update.
CommFail	logical	yes	no	Goes high if Lookout*Direct* cannot communicate with the server.
DataError	logical	yes	no	Goes high if the object cannot properly process the data returned by the server.

Table 3-15. OPCClient Data Members (Continued)

Data Member	Type	Read	Write	Description
PollDevice	logical	no	yes	When transitioned from FALSE to TRUE, the object polls the server.
Update	logical	yes	no	Pulses high and low when the OPC server updates Lookout*Direct* or Lookout*Direct* successfully polls the server.

The data members for the OPC Client depend on your OPC server. To use one of the data members from your server, enter the data member directly into an expression or URL field.

If the OPC server permits browsing, you can use data members as you would with any other Lookout*Direct* driver object. If you cannot browse your OPC server, enter the item manually in the form acceptable to your OPC server, as in the following example:

`OPCclient1.'`*`device`*`\`*`folder`*`\`*`itemName`*`'`

Note Notice the use of single quotes. Using a single quote around a component of a path allows the use of any character in that path, not just the normally allowed characters.

If all your items use the same access path, use the **Default Access Path** option. If you need to specify a different access path, use a period and tilde (.~) to denote the first element of the path, as in the following example:

`OPCclient1.'`*`device`*`\`*`itemN`*`'.'~`*`accesspath`*`'`

Notice that the access path must be a separate component. Single quotes denote a string as being a single component.

If an item name or access path already contains a tilde, you must enter one additional tilde to act as an escape character for that tilde, and a second additional tilde to denote an access path (for example, if the access path is `~COM1`, enter `~~~COM1`).

Examples

```
OPCclient1.'4:0'.'~Modbus Demo Box'
```

The access path is `Modbus Demo Box` and the OPC item ID is `4:0`.

```
OPCclient1.'4:0'
```

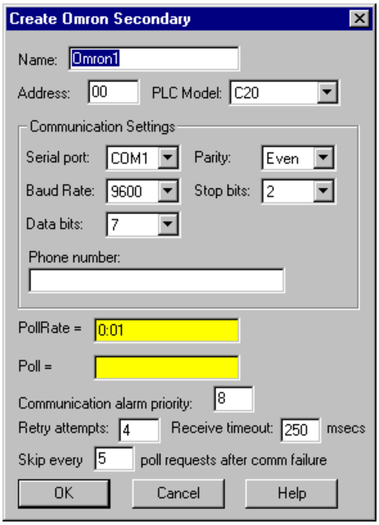
The access path is Null and the OPC item ID is `4:0`.

Note You cannot browse access paths for an item. You must enter access paths manually.

Omron

Omron is a protocol driver class Lookout*Direct* uses to communicate with Omron devices using the Host Link serial communication protocol.

An Omron object contains a great deal of data. It supports reading and writing of all predefined data points. When you create an Omron object, you have immediate access to all the data members for that object (see *Omron Data Members* list in Table 3-16).



Create Omron Secondary

Name: Omron1

Address: 00 PLC Model: C20

Communication Settings

Serial port: COM1 Parity: Even

Baud Rate: 9600 Stop bits: 2

Data bits: 7

Phone number:

PollRate = 0:01

Poll =

Communication alarm priority: 8

Retry attempts: 4 Receive timeout: 250 msecs

Skip every 5 poll requests after comm failure

Serial port specifies which COM port the object uses for communicating to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports…** command.

Data rate indicates the baud rate that Lookout*Direct* uses to communicate with the hardware device. This **Data rate** setting should match the selection made on the physical device.

Data bits indicates the number of data bits that Lookout*Direct* uses to communicate with the hardware device. This **Data bits** setting should match the selection made on the physical device.

Stop bits indicates the number of stop bits that Lookout*Direct* uses to communicate with the hardware device. This **Stop bits** setting should match the selection made on the physical device.

Parity indicates the parity that Lookout*Direct* uses to communicate with the hardware device. This **Parity** setting should match the selection made on the physical device.

Phone number specifies the number to be dialed if the selected serial port is configured for dial-up. This **Phone number** only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. The object then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *How LookoutDirect Works*, of the *Getting Started with LookoutDirect* manual for more information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout*Direct* polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of alarms generated by the Omron object. Such alarms are typically related to communications with the physical device.

Retry attempts specifies the consecutive number of times Lookout*Direct* attempts to establish communications with a device when it is not getting a valid response. After it tries the number of **Retry attempts** specified, the Omron object generates an alarm and releases the communication port back to the communications subsystem. The subsystem then moves on to the next device in the polling queue (if any). See Chapter 3, *Serial Communications*, in the *LookoutDirect Developer's Manual* for more information.

Receive timeout is the time delay Lookout*Direct* uses in waiting for a response from a device before retrying the request.

The **Skip every…** setting instructs Lookout*Direct* not to poll a device it has lost communication with on every scheduled poll. Instead, Lookout*Direct* skips the device in the polling cycle. Once communications have been reestablished, the device is polled on its regular cycle.

Omron Data Members

As with all Lookout*Direct* drivers, you can access I/O points and other data through data members. The following is a table of data members currently supported by the Omron object class.

Table 3-16. Omron Data Members

Data Member	Type	Read	Write	Description
AR0–AR27	numeric	yes	yes	Auxiliary relay area, read as 16-bit word.
AR0.0–AR27.15	logical	yes	yes	Auxiliary relay area, read as 1-bit discrete.
DM0–DM9999	numeric	yes	yes	Data memory area, 16-bit word.
CommFail	logical	yes	no	Object-generated signal that is on if, for any reason, Lookout*Direct* cannot communicate with the device(s).
HR0–HR99	numeric	yes	yes	Holding relay area, read as 16-bit word.
HR0.0–HR99.15	logical	yes	yes	Holding relay area, read as 1-bit discrete.
IR0–IR511	numeric	yes	yes	I/O area, read as 16-bit word.
IR0.0–IR511.15	logical	yes	yes	I/O area, read as 1-bit discrete.
LR0–LR63	numeric	yes	yes	Link relay area, read as 16-bit word of information.
LR0.0–LR63.15	logical	yes	yes	Link relay area, read as 1-bit discrete.
TC0–TC999	numeric	yes	no	Timer/Counter, read as 16-bit word.
Update	logical	yes	no	Object-generated signal that pulses low each time it polls the device.

Note The Omron requires a special cable configuration in order to work properly. See your Omron hardware documentation for the correct configuration.

Omron Status Messages

No response within timeout period

Lookout*Direct* received no response from a device within the **Receive timeout** period. The Omron object is able to use the COM port, but when it polls the device, it does not respond—as if it is not even there.

Cannot set PLC to MONITOR mode

The Omron object is trying to set the PLC in MONITOR mode in order to communicate with the PLC correctly, but cannot perform the operation.

Incorrect address in response

The frame received had an incorrect source address. Check for two or more devices with the same address.

Incorrect command in response

The frame received had an incorrect command. Check for two or more devices with the same address.

Incorrect data type in response

The frame received had an incorrect data type marker.

Incorrect frame check sum (FCS)

The frame received had an incorrect check sum.

Omron errors reported in the response

These errors are reported by the Omron device, and are in turn reported to you in text form.

Omron Models Supported

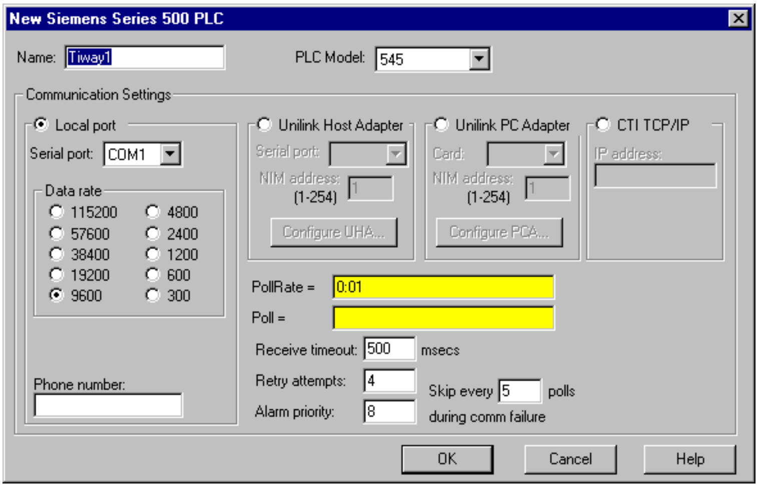
C20, C200, C500, C1000, C2000, CQM, CPM1

Tiway

Tiway is a protocol driver object Lookout uses to communicate with series 5*xx* PLCs manufactured by Siemens, formerly made by Texas Instruments.

Protocol driver objects contain a great deal of data. All readable and writable members (inputs/outputs), polling instructions, read/write blocking, serial port usage, and so on are bundled with the object. As soon as you create a Tiway object you immediately have access to all the object data members (see data member list in this section).

Note Lookout protocol driver objects automatically generate an efficient read/write blocking scheme based on the inputs and outputs being used in your process file. You are not required to build your own I/O blocking table.



Name: Tiway1

PLC Model: 545

● Local port

Serial port: COM1

Data rate

○ 115200	○ 4800
○ 57600	○ 2400
○ 38400	○ 1200
○ 19200	○ 600
● 9600	○ 300

PollRate = 0:01

Poll =

Receive timeout: 500 msecs

Retry attempts: 4

Alarm priority: 8

Skip every 5 polls during comm failure

PLC Model specifies the PLC model number for the requested device.

PollRate is a numeric expression that determines how often to poll the device. Tiway then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *How Lookout Works*, of the *Getting Started with Lookout* manual for more information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device it does not get a valid response from. After Retry attempts times, Tiway generates a communication alarm and Lookout moves on to the next device in the polling queue (if any).

Alarm priority determines the priority level of Tiway generated alarms.

Phone number specifies the number to be dialed if the serial port setting is configured for dial-up. This number only applies to the individual protocol object.

The **Skip every ___ polls** setting instructs Lookout not to poll a device it has lost communication with on every scheduled poll. Instead, Lookout skips the device in the polling cycle accordingly. Once communications have been reestablished, the device is polled on its regular cycle.

Update Write Settings

The Lookout default for the Tiway driver object is for the object to perform an update write to the PLC registers every 100 polls. Notice that this can be problematic if the PLC has been changing its own register values. To change this default operation, you must create an entry in the `Lookout.INI` file. Create a Tiway group with the key UpdateOutputs.

Setting the UpdateOutputs key equal to 0 means the Tiway object will not perform update writes. Setting the key equal to some positive integer N will set the Tiway object to perform update writes once every N polls.

See Appendix C, *The Lookout .INI File*, in the *Lookout Developer's Manual* for more information on using the `Lookout.INI` file. You can also refer to the `Lookout.INI` file topic in Lookout Help for additional information.

Communication Techniques

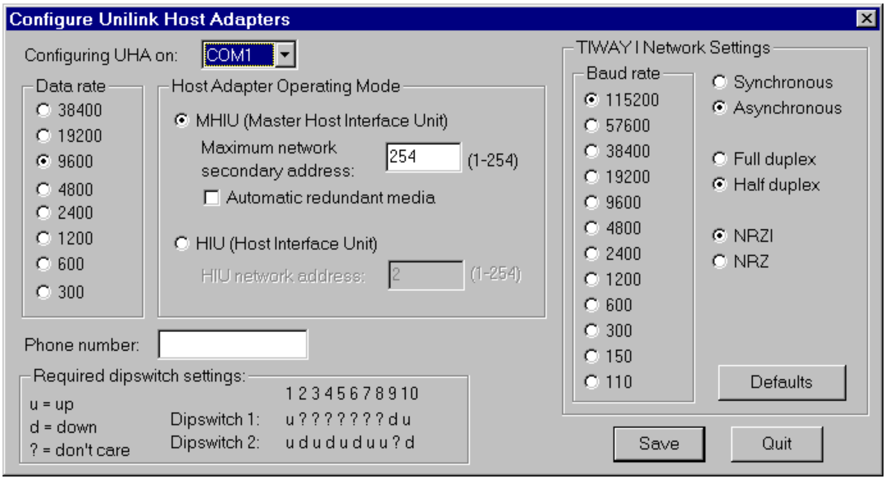
Lookout communicates with Siemens PLCs in several ways: direct serial connection to the **Local port**, serial connection to an **external Unilink Host Adapter**, through an internal **Unilink PC Adapter** card, or through an internal **CTI TCP/IP** card.

Local Port

The **Local port** settings determine the **serial port**, **data rate**, and **phone number** (if any) to be used in a direct connect setup. Because the **Local port** protocol does not include address information, this option is limited to only one (1) PLC per serial port.

Unilink Host Adapter

If **Unilink Host Adapter** is selected, you must specify the **Serial port** to be used and the **NIM** (Network Interface Module) **address** as set at the PLC. You also should configure several settings on the Unilink Host Adapter by selecting the **Configure UHA…** button.



Configure Unilink Host Adapters

Configuring UHA on: COM1

Data rate
○ 38400
○ 19200
● 9600
○ 4800
○ 2400
○ 1200
○ 600
○ 300

Host Adapter Operating Mode
● MHIU (Master Host Interface Unit)
Maximum network secondary address: 254 (1-254)
□ Automatic redundant media
○ HIU (Host Interface Unit)
HIU network address: 2 (1-254)

Phone number:

Required dipswitch settings:
u = up
d = down
? = don't care

```
              1 2 3 4 5 6 7 8 9 10
Dipswitch 1:  u ? ? ? ? ? ? ? d u
Dipswitch 2:  u d u d u d u u ? d
```

TIWAY I Network Settings

Baud rate
● 115200
○ 57600
○ 38400
○ 19200
○ 9600
○ 4800
○ 2400
○ 1200
○ 600
○ 300
○ 150
○ 110

○ Synchronous
● Asynchronous

○ Full duplex
● Half duplex

● NRZI
○ NRZ

The settings in this dialog box are globally applied to all PLCs on the specified TIWAY network (each network requires a separate serial port). Therefore, it is only necessary to configure each Unilink Host Adapter one time—you need not repeat this step every time you create a new Tiway object.

Data rate specifies the communication speed between the computer and the Unilink Host Adapter. It also determines the required dip switch settings on the UHA for the selected baud rate.

The **Host Adapter Operating Mode** determines if the Unilink Host Adapter is the network manager (**Master Host Interface Unit**) or just another network secondary (**Host Interface Unit**). There must be exactly one MHIU per TIWAY network.

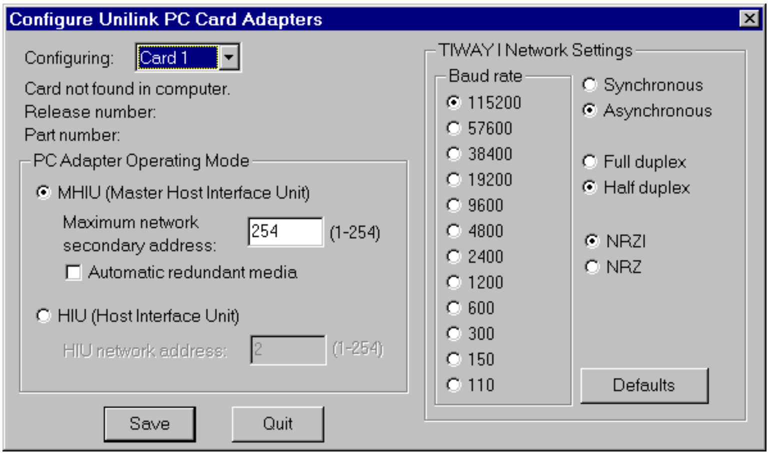
Enabling **Automatic redundant media** instructs the Unilink Host Adapter to attempt communications over a redundant TIWAY network to any secondary it loses communications with.

The **TIWAY I Network Settings** configure the communication parameters for the TIWAY network. This network runs between the Unilink Host Adapter and its secondaries. The Lookout default network settings correspond to the default NIM settings as shipped from Siemens. See your TIWAY documentation to modify any of these parameters.

Unilink PC Adapter

Because the **Unilink PC Adapter** is an internal card, it eliminates the 19200 baud serial bottleneck and replaces it with the 8 MHz PC ISA bus speed. Therefore, the performance gains over the **Unilink Host Adapter** and **Local port** settings can be substantial.

If **Unilink PC Adapter** is selected, you must specify the **Card** to be used and the **NIM** (Network Interface Module) **address** as set at the PLC. You should also configure several settings on the Unilink PC Adapter by selecting the **Configure PCA…** button.



Configure Unilink PC Card Adapters

Configuring: Card 1

Card not found in computer.
Release number:
Part number:

PC Adapter Operating Mode

● MHIU (Master Host Interface Unit)

Maximum network secondary address: 254 (1-254)

□ Automatic redundant media

○ HIU (Host Interface Unit)

HIU network address: 2 (1-254)

TIWAY I Network Settings

Baud rate

● 115200
○ 57600
○ 38400
○ 19200
○ 9600
○ 4800
○ 2400
○ 1200
○ 600
○ 300
○ 150
○ 110

○ Synchronous
● Asynchronous

○ Full duplex
● Half duplex

● NRZI
○ NRZ

The settings in this dialog box are globally applied to all PLCs on the specified TIWAY network (each network requires a separate card). Therefore, it is only necessary to configure each Unilink PC Adapter one time—this step need not be repeated every time a new Tiway object is created.

The **PC Adapter Operating Mode** determines if the Unilink PC Adapter is the network manager (**Master Host Interface Unit**) or just another network secondary (**Host Interface Unit**). There must be exactly one MHIU per TIWAY network.

Enabling **Automatic redundant media** has no effect with the PC Adapter card because it has only one port. If Siemens adds a second port, Lookout automatically supports this option.

The **TIWAY I Network Settings** configure the communication parameters for the TIWAY network. This network runs between the Unilink PC Adapter card and its secondaries. Lookout default network settings correspond to the default NIM settings as shipped from Siemens. See your TIWAY documentation to modify any of these parameters.

CTI TCP/IP

Lookout supports the Control Technology Incorporated (CTI) Ethernet TCP/IP adapter cards that can be installed in SIMATIC TI545 PLCs. In order to work with such cards, your PC must be equipped with an Ethernet network card and a Windows Sockets-Compliant TCP/IP software package. Such packages are available from Microsoft, FTP Software, and NetManage, Inc.

The Lookout **CTI TCP/IP** protocol option is Windows Sockets Compliant. It uses connectionless UDP sockets in software, an industry standard for TCP/IP protocols. In this protocol, a FIFO (first-in, first-out) stack is used to temporarily store communication messages if the data highway is busy or if multiple poll request are generated by several Tiway objects.

Because **CTI TCP/IP** utilizes sockets to momentarily store poll requests, this protocol eliminates bottlenecks imposed by multiple Tiway objects trying to access the data highway at the same time. Performance gains over **Local port**, **Unilink Host Adapter** and **Unilink PC Adapter** settings can be substantial when you are configuring a system that has several PLCs on the same network.

If **CTI TCP/IP** is selected, you need to specify the **IP address** (Internet protocol address) of the PLC. An Internet protocol address consists of four numbers, separated by periods. Each number ranges from zero to 255 decimal. Thus, a typical Internet address might be 128.7.9.231. Ensure that the **IP address** you enter matches the Internet protocol address of the PLC as specified in its EEPROM or as programmed using PCL.

You can add a secondary IP address to the **CTI TCP/IP** parameter. Lookout now toggles between the primary and secondary IP address after a COM failure (assuming a secondary address exists). Enter the secondary ID after the first, preceded by a space or a comma. For example:

```
207.68.156.61, 1.2.3.4
```

Tiway Data Members

Table 3-17. Tiway Data Members

Data Members	Type	Read	Write	Description
AERR1 – AERR32000	numeric	yes	yes	(Analog Alarm) Error
AHA1 – AHA32000	numeric	yes	yes	(Analog Alarm) High alarm limit
AHHA1 – AHHA32000	numeric	yes	yes	(Analog Alarm) High high alarm limit
ALA1 – ALA32000	numeric	yes	yes	(Analog Alarm) Low alarm limit
ALLA1 – ALLA32000	numeric	yes	yes	(Analog Alarm) Low low alarm limit
AODA1 – AODA32000	numeric	yes	yes	(Analog Alarm) Orange deviation limit
APVH1 – APVH32000	numeric	yes	yes	(Analog Alarm) Process variable high limit
APVL1 – APVL32000	numeric	yes	yes	(Analog Alarm) Process variable low limit
ARCA1 – ARCA32000	numeric	yes	yes	(Analog Alarm) Rate of change limit
ASP1 – ASP32000	numeric	yes	yes	(Analog Alarm) Setpoint
ASPH1 – ASPH128	numeric	yes	yes	(Analog Alarm) Setpoint high limit
ASPL1 – ASPL128	numeric	yes	yes	(Analog Alarm) Setpoint low limit
ATS1 – ATS32000	numeric	yes	yes	(Analog Alarm) Sample rate
AVF1 – AVF128	numeric	yes	yes	(Analog Alarm) Alarm flags
C1 – C32000	logical	yes	yes	Control Registers
CommFail	logical	yes	no	Driver-generated signal that is ON if Lookout cannot communicate with the device for whatever reason

Table 3-17. Tiway Data Members (Continued)

Data Members	Type	Read	Write	Description
K1 – K32000	numeric	yes	yes	K-memory unsigned 16-bit integer value ranging from 0 to 65535
K1. – K32000.	numeric	yes	yes	K-memory 32-bit IEEE floating point value
K1D – K32000D	numeric	yes	yes	K-memory 32-bit unsigned integer value
K1S – K32000S	numeric	yes	yes	K-memory signed 16-bit integer value ranging from –32768 to 32767
LADB1 – LADB64	numeric	yes	yes	(Analog Alarm) Deadband
LADB1 – LADB64	numeric	yes	yes	(Loop) Deadband
LER1 – LER64	numeric	yes	no	(Loop) Error
LHA1 – LHA64	numeric	yes	yes	(Loop) High alarm limit
LHHA1 – LHHA64	numeric	yes	yes	(Loop) High high alarm limit
LKC1 – LKC64	numeric	yes	yes	(Loop) Gain
LKD1 – LKD64	numeric	yes	yes	(Loop) Derivative gain
LLA1 – LLA64	numeric	yes	yes	(Loop) Low alarm limit
LLLA1 – LLLA64	numeric	yes	yes	(Loop) Low low alarm limit
LMN1 – LMN64	numeric	yes	yes	(Loop) Output
LMX1 – LMX64	numeric	yes	yes	(Loop) Bias
LODA1 – LODA64	numeric	yes	yes	(Loop) Orange deviation limit
LPV1 – LPV64	numeric	yes	yes	(Loop) Process variable
LPVH1 – LPVH64	numeric	yes	yes	(Loop) Process variable high limit
LPVL1 – LPVL64	numeric	yes	yes	(Loop) Process variable low limit
LRCA1 – LRCA64	numeric	yes	yes	(Loop) Rate of change limit
LSP1 – LSP64	numeric	yes	yes	(Loop) Setpoint
LSPH1 – LSPH64	numeric	yes	yes	(Loop) Setpoint high limit
LSPL1 – LSPL64	numeric	yes	yes	(Loop) Setpoint low limit

Table 3-17. Tiway Data Members (Continued)

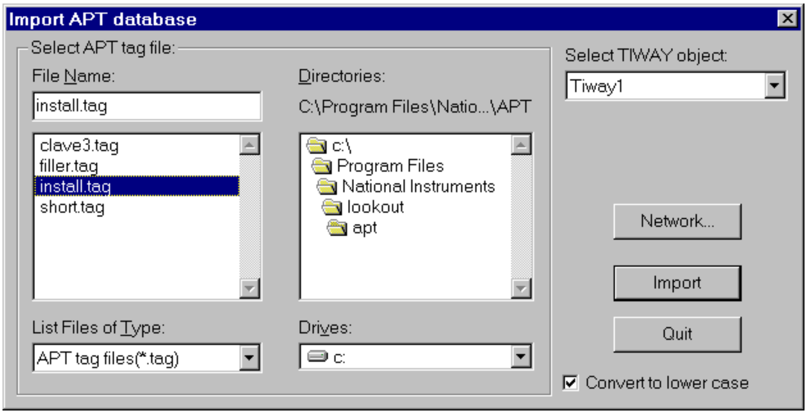
Data Members	Type	Read	Write	Description
LTD1 – LTD64	numeric	yes	yes	(Loop) Rate
LTI1 – LTI64	numeric	yes	yes	(Loop) Reset
LTS1 – LTS64	numeric	yes	yes	(Loop) Sample rate
LYDA1 – LYDA64	numeric	yes	yes	(Analog Alarm) Yellow deviation limit
LYDA1 – LYDA64	numeric	yes	yes	(Loop) Yellow deviation limit
Poll	logical	no	yes	When this value transitions from FALSE to TRUE, Lookout polls the device
PollRate	numeric	no	yes	Specifies the frequency at which the Lookout object polls the device
STW1 – STW32000	numeric	yes	no	Status Words
TCC1 – TCC32000	numeric	yes	yes	(Analog Alarm) Timer/counter current
TCP1 – TCP32000	numeric	yes	yes	(Analog Alarm) Timer/counter preset
Update	logical	yes	no	Driver-generated signal that pulses each time the driver polls the device
V1 – V32000	numeric	yes	yes	V-memory unsigned 16-bit integer value ranging from 0 to 65535
V1. – V32000.	numeric	yes	yes	V-memory 32-bit IEEE floating point value
V1B1 – V32000B16	logical	yes	yes	One bit of a word written out as a whole word
V1D – V32000D	numeric	yes	yes	V-memory 32-bit unsigned integer value
V1S – V32000S	numeric	yes	yes	V-memory signed 16-bit integer value ranging from –32768 to 32767
V1T – V32000T	text	yes	yes	Two characters of text

Table 3-17. Tiway Data Members (Continued)

Data Members	Type	Read	Write	Description
WX1 – WX32000	numeric	yes	no	Word Image Inputs—16-bit values that typically range from 6400 – 32000 for 4 – 20 mA signals, and 0 – 32000 for 0 – 5V signals.
WY1 – WY32000	numeric	yes	yes	Word Image Outputs—16-bit values that typically range from 6400 – 32000 for 4 –20 mA signals, and 0 – 32000 for 0 – 5V signals.
X1 – X32000	logical	yes	no	Discrete Inputs—unassigned Xs may be used as control registers
Y1 – Y32000	logical	yes	yes	Discrete Outputs—same memory space as Discrete Inputs, so X37 references the same point as Y37. Unassigned Ys may be used as control registers

Importing APT Name Files

After you have created at least one Tiway object, the Tiway class adds a menu selection to the Lookout **Options** menu you can use to import an APT name file database for each Tiway object created. You can re-import name files as your APT programs are modified, and Lookout readjusts the aliased name names automatically, in real time.


Import APT database

Select APT tag file:

File Name: install.tag

clave3.tag
filler.tag
install.tag
short.tag

Directories: C:\Program Files\Natio...\APT

c:\
Program Files
National Instruments
lookout
apt

List Files of Type: APT tag files(*.tag)

Drives: c:

Select TIWAY object: Tiway1

☑ Convert to lower case

Glossary

Prefix	Meanings	Value
m-	milli-	10^{-3}
k-	kilo-	10^{3}
M-	mega-	10^{6}

A

absolute date
absolute time

Numeric system Lookout*Direct* uses for keeping track of dates and times, in which midnight (0 hours), January 1, 1900 is represented by 1, midnight of January 2, 1900 is represented by 2, and so on. The absolute date/time number 36234.47222250 represents 11:20 A.M., March 15, 1999.

The numeric value for 1 second in Lookout*Direct* is .000011574, the numeric value for 1 minute is .000694444, and the numeric value for 1 hour is .041666667.

ACK

Acknowledge (an alarm or event).

active notification

A feature of event-driven software systems in which the application is alerted of value changes when they occur instead of through continuous, loop-driven queries.

address space

An OPC term for the area you browse to find what items are available on an OPC server. Part of the standard OPC interface, this space may arrange items hierarchically.

alarm

Software notification of a condition in a process. This alarm may call attention to a value that has exceeded or fallen below certain levels, set in the object database or in an Alarm object.

alias

Name given to a data member using the **Edit Database** dialog box. This name can be descriptive or mnemonic, and can be associated with other data member configurations such as scaling, logging, and alarming. A data member can have more than one alias, each with different associated configurations.

B

baud rate

Measurement of data transmission speed, formally defined as the number of electronic state changes per second. Because most modems transmit four bits of data per change of state, is sometimes misused or misunderstood—a 300 baud modem is moving 1200 bits per second. *See* bps.

.bmp files

Graphic files in bitmap format. If you are using a `.BMP` file in Lookout*Direct*, you cannot resize it on screen. *See* Windows metafile.

bps

Bits per second—measure of the rate of transfer of data.

C

CBL compiler

Lookout*Direct* uses the CBL (Control Block Language) compiler to compile a Lookout*Direct* source file (`.lks`) into a binary file (`.l4p`).

.cbx file

A Lookout*Direct* file containing a Lookout*Direct* object class. A `.CBX` (Control Block Extension) file can have one or more object classes in it.

checksum

A method of verifying that the number of bits received is the same as the number of bits transmitted. Used by TCP/IP and serial protocols.

Citadel

The Lookout*Direct* historical database that stores your data for access later.

classes

See object classes.

client

A Lookout*Direct* process that monitors a Lookout*Direct* server process. Lookout*Direct* clients should be computer independent so that they can be run from any computer on your network. Lookout*Direct* server processes run on computers actually connected to your control hardware.

comm port

Term sometimes used for a serial port.

connection

Input to a Lookout*Direct* object's writable data members. For more information, refer to Chapter 4, *Using LookoutDirect*, in your *Getting Started with LookoutDirect* manual.

control objects

Lookout*Direct* objects you use to control a process, change a data value, adjust a register, and so on.

controllable objects

Lookout*Direct* objects you can control with a Lookout*Direct* control object.

.csv files

Comma Separated Value file, a format widely accepted by spreadsheet and other data handling programs.

CTS

Clear to Send. Part of a handshaking protocol for certain devices that connect the serial port of a computer. See the *RTS/CTS Handshaking Settings* section of Chapter 3, *Serial Port Communication Service,* for detailed information.

cursor (data table)

The Lookout*Direct* data table can activate one row of data at a time using the data table cursor. See the Data Table reference in the online help or the Lookout*Direct Object Reference Manual*.

D

DAQ

Short for Data AcQuisition.

data member

Data source or sink associated with a Lookout*Direct* object. A readable data member, or source, can be used in expressions or as inputs to other objects. A writable data member, or sink, can have at most one connection into it, created using the **Object»Edit Connections** dialog box. A data member can be both readable and writable.

See also native data member and *alias*.

data type

Kind of value (numeric, logical, or text) that a parameter or data member can hold.

database

Collection of data stored for later retrieval, display, or analysis.

datagram

Message sent between objects in Lookout*Direct*. A datagram contains a route and a value.

DCOM/COM

Distributed Component Object Model, a Microsoft standard in which client program objects request services from server program objects.

The Component Object Model (COM) is a set of interfaces, clients, and servers used to communicate within the same computer (running Windows 98/95 or Windows NT).

DDE

Dynamic Data Exchange, currently used in Lookout*Direct* to exchange data with other programs (such as Microsoft Excel) running on your network.

deadband

A value that must be exceeded for an alarm to sound or a change in state to be recorded. For instance, if you have a low-level alarm set at 5 with a deadband of 2, the alarm will not trigger until the value being monitored drops to 5. The alarm will then stay active until the value being monitored moves above 7. A deadband keeps small oscillations of value from triggering an alarm and then canceling it too rapidly.

deviation

Set a deviation to filter out small changes in value when logging data. Before being logged to a database, a value must change by at least the deviation amount of the last logged value.

dialing prefix

Part of the Hayes AT command set for use with modems. See the *Dial-Up Modem Settings* section of Chapter 3, *Serial Port Communication Service*, for detailed information.

displayable objects

A Lookout*Direct* object class that has a displayable component, such as a Pot, a Switch, or a Pushbutton.

DLL

Dynamic Link Library, which is a collection of small, special-purpose programs which can be called by a larger program running on the computer. Sometimes called Dynamically Linked Library.

driver objects

Lookou*Direct* objects used to communicate with PLCs, RTUs, and other I/O devices.

E

edit mode

Lookout*Direct* mode in which you can alter and create objects within a process. Switch in and out of edit mode by pressing <Ctrl-space> or by selecting **Edit»Edit Mode**.

engineering unit

In Lookout*Direct*, used to refer to scaled or converted data. Thermocouple data, for instance, arrives in volts as the raw unit, and must be converted to degrees, an engineering unit.

environment services

Tasks Lookout*Direct* performs as a part of making your SCADA/HMI work easier. Lookout*Direct* environment services include serial communications, database and logging, security, networking, alarming, and so on.

Ethernet

A widely used, standardized local area networking technology, specified in the IEEE 802.3 standard.

event

Anything that happens can be an event. In Lookout*Direct*, events include such things as adjusting a control value, entering or exiting edit mode, opening or closing a control panel, and logging in or logging out of the system.

expression functions

Mathematical, logical, and other functions used by Lookout*Direct* expressions.

expressions

Lookout*Direct* expressions are often paths to a data member. They can also function like variables that, using a spreadsheet cell-type formula, become capable of performing flexible, real-time math operations, condition testing, and other complex operations functions. See Chapter 1, *Expressions*, for more information on expressions.

F

failover

A failover is the takeover of a process by a standby computer when the primary computer fails for any reason.

FieldBus

An all-digital communication network used to connect process instrumentation and control systems.

FieldPoint

A National Instruments hardware product line for industrial automation, control, monitoring, and reporting.

frame

Sequence of bytes sent from a computer to a device or vice versa. The syntax of the frame depends on the protocol being used. A read frame contains enough information to specify a set of variables whose values the device should return. A write frame specifies a variable in the device and a new value to write into that variable. Some protocols support the writing of multiple variables in a single frame. A response frame is returned from the device to the computer, indicating whether the frame just sent to it was received successfully. If the frame just received was a read frame, the response frame contains a set of requested values.

functionality

The way an object works, operates, or performs a task. Functionality is a general concept that applies in the same way to all objects in a given object class. Parameters define the specific functionality of an individual object.

functions

See expression functions.

G

gray proximity

A term used in Lookout*Direct* color animation. This sets what percentage of gray will be replaced by a given color as conditions change in a monitored value or set of values.

H

Hi and HiHi

Alarm settings. Both warn that a value has gone above some setpoint. Generally a Hi alarm is used to alert an operator of a need for intervention. A HiHi alarm is usually used to alert an operator that the value has been exceeded by an even greater margin than a Hi alarm indicates, and is usually used to indicate an urgent need for action.

historical logging

The process of storing data in a database for use at another time, or from another location.

HOA

Hand-Off-Auto control, used to set whether a value must be changed manually, is completely turned off, or functions automatically. You can use a Pot object and a complex expression to create this sort of control in LookoutDirect, or you can use a RadioButton object, depending on the particular requirements of the task you need to accomplish.

I

I/O point

Every read-only, write-only, or read-write connection Lookout*Direct* makes to external hardware is counted as an I/O point. Lookout*Direct* is licensed for use with a set number of I/O points. If you exceed the number you are licensed to use with your copy of Lookout*Direct*, a warning message appears on your computer screen warning you to shut down one of your processes within a specified time before Lookout*Direct* cuts back on I/O usage.

(implicit) data member

A Lookout*Direct* data member containing the fundamental data for certain object classes. When you make a connection to an (implicit) data member, you only use the name of the object, not the name of the object followed by the data member name.

L

.l4p files

File extension for Lookout*Direct* process files. These are the compiled files LookoutDirect runs when it runs a process.

.l4t files

File extension for a Lookout*Direct* state file, which stores the values for Lookout*Direct* controls and other objects with state information.

.lka files

File extension for Lookout*Direct* security files.

.lkp files

File extension for Lookout*Direct* process files in versions of Lookout*Direct* earlier than Lookout*Direct* 4.

.lks files

File extension for a Lookout*Direct* source file, which Lookout*Direct* compiles to make a Lookout*Direct* process file that Lookout*Direct* can run. This is the file you should make sure you keep backed up in case you need to recreate a corrupted process file, or in case some future version of Lookout*Direct* cannot run a process file compiled in an earlier version of Lookout*Direct*.

logging

The process of storing data in a computer database file. See Chapter 7, *Logging Data and Events*, for more information on logging data in Lookout*Direct*.

logical data member

A Lookout*Direct* data member of the logical data type.

.lst files

Extension for the Lookout*Direct* state file in versions of Lookout*Direct* earlier than Lookout*Direct* 4.

M

multiplex

A method of working with more than one data stream using only one communications channel. There are a number of different methods of multiplexing, depending on the hardware and software being used. A number of Lookout*Direct* driver objects support multiplexing hardware.

N

native data member

Data members built into a Lookout*Direct* object class, as opposed to data members you create by using aliases.

NetDDE

A way of networking using DDE (dynamic data exchange), retained in Lookout*Direct* 4 for compatibility with earlier versions of Lookout*Direct*.

numeric data member

A Lookout*Direct* data member of the numeric data type.

O

object

A specific instance created from an object class.

object classes

Software modules you use to create individual objects to perform tasks in Lookout*Direct*.

object connections

Software links between objects used to transmit data and commands from one object to another.

ODBC

Open DataBase Connectivity, a standard application programming interface (API) for accessing a database. You can use ODBC statements to access files in a number of different databases, including Access, dBase, DB2, and Excel.

ODBC is compatible with the Structured Query Language (SQL) Call-Level Interface. ODBC handles SQL requests by converting them into requests an ODBC database can use.

OPC

OLE for Process Control, an industry standard interface providing interoperability between disparate field devices, automation/control systems, and business systems. Based on ActiveX, OLE, Component Object Model (COM), and Distributed COM (DCOM) technologies.

P

parameter

Input to an object, similar to a writable data member, whose value is specified in the object parameter list in a Lookout*Direct* source (`.LKS`) file. Typically, parameter values are set in the object **Object»Create** or **Object»Modify** dialog box.

ping

A small utility program in Windows and DOS that checks to see if a computer can be reached across a network. Also used to indicate the running of that program.

pixel

Picture Element, the smallest bit of a picture. Has one color or shade of grey. The number of pixels per inch determine the resolution of an image.

PLC

Programmable Logic Controller.

poll

A software event in which a computer checks some value in a device or register. In Lookout*Direct*, a logical command that forces a device poll to check data member values.

poll rate

How often a device is polled.

pop-up panel

One variety of Lookout*Direct* control panel that can only be displayed at the size set by the process developer, and which cannot be maximized. When open, a popup panel remains on top of other panels until minimized.

process

In Lookout*Direct*, process refers to a Lookout*Direct* “program”, used for industrial automation, control, monitoring, or reporting.

process file

The Lookout*Direct* binary file Lookout*Direct* executes when running a process. Carries the `.l4p` extension.

R

raw unit

Data as it arrives in your process, such as voltage or amperage. Thermocouple data, for instance, arrives in volts as the raw unit, and must be converted to degrees, an engineering unit.

receive gap

A serial communications setting that determines the number of empty bytes (or amount of time) a driver receives before recognizing the end of a message frame and requesting another message. See the *Setting Receive Gap* section of Chapter 3, *Serial Port Communication Service*, for more information about the receive gap.

redundancy

A system for making sure that a computer can come online and run a Lookout*Direct* process if the computer currently running that process fails for some reason.

remote

In the context of Lookout*Direct*, remote is a position source location for a control. See the *Remote Position Source* section of Chapter 4, *Using LookoutDirect,* in the *Getting Started with LookoutDirect* manual for detailed information on the Lookout*Direct* remote position source.

resolution

The smallest signal increment that can be detected by a measurement system. Also, the number of pixels per inch on a computer monitor screen or dots per inch in printer output.

RTS

Request to Send, part of a handshaking protocol for certain devices that connect the serial port of a computer. See the *RTS/CTS Handshaking Settings* section of Chapter 3, *Serial Port Communication Service*, for detailed information.

RTU

Remote Terminal Unit, a device similar to a PLC for use at a remote location, communicating with a host system through radio or telephonic connections.

run mode

Lookout*Direct* mode in which processes run but no editing changes can be made. Switch in and out of run mode by pressing <Ctrl-space> or selecting **Edit»Edit Mode**.

S

SCXI

Signal Conditioning eXtensions for Instrumentation, a National Instruments product line for conditioning low-level signals.

security accounts

Also called user and group accounts, Lookout*Direct* uses security accounts to define what users or group of users have different operation privileges in Lookout*Direct*. See Chapter 6, *Security*, for detailed information on Lookout*Direct* security.

server

A process that provides data (services) to client processes. In Lookout*Direct*, server processes are intended to be run on one computer only, with direct connections to field hardware. Client processes interact with field hardware through server processes.

source file

Lookou*Direct*t file that can be compiled to produce a binary Lookout*Direct* process file that runs a process. Uses a `.lks` file extension.

SQL

Structured Query Language, used to get information from and update information in a database.

standby

A computer standing by to take over running a process if the primary computer fails or falls offline.

startup file

A Lookout*Direct* process file (`.l4p`) you designate in the **System Options** dialog box that Lookout*Direct* will open and run any time Lookout*Direct* is opened.

state file

The Lookout*Direct* file that stores the value of all Lookout*Direct* control parameters and object data members in use in a process. Uses the file extension `.l4t`.

system objects

Lookout*Direct* objects used to control other objects or process and analyze data.

T

TCP
TCP/IP

Transmission Control Protocol, a method (protocol) for sending data between computers. Used with IP, the Internet Protocol.

TCP/IP sends data as packets, with IP handling the delivery of data and TCP keeping track of the individual packets.

text data member

Lookout*Direct* data member used for text data.

trace

A term for data from a single source over some period of time, stored in an ODBC-compliant database.

traces table

ODBC databases present data in the form of traces tables. A traces table contains a field or column of data for each data member being logged, along with a field you can use to query the database.

trend

Historical data showing the change in a value over time. Often used in connection with graphing the data for display.

W

.wav files

File extension given to sound files. You can play a `.wav` file in Lookout*Direct* to add sounds or speech to alarms or events.

Windows metafile

A standard graphics file type for use in the Microsoft Windows operating environment. If you use a metafile graphic in Lookout*Direct*, you can enlarge or reduce it on the screen, use them as masks without specifying transparent pixels, and use the Lookout*Direct* Animator to animate the colors of the graphic.

.wmf files

File extension given to Windows Metafile graphic files.

X

.xls files

File extension given to Microsoft Excel files.

Index