

FactoryLink 6.5.0

.....

Fundamentals

-
-
-
-

©Copyright 1984 - 1998 United States Data Corporation. All rights reserved.

- NOTICE -

The information contained herein is confidential information of United States Data Corporation, a Delaware corporation, and is protected by United States copyright and trade secret law and international treaties. This document may refer to United States Data Corporation as "USDATA."

Information in this document is subject to change without notice and does not represent a commitment on the part of United States Data Corporation ("USDATA"). Although the software programs described in this document (the "Software Programs") are intended to operate substantially in accordance with the descriptions herein, USDATA does not represent or warrant that (a) the Software Programs will operate in any way other than in accordance with the most current operating instructions available from USDATA, (b) the functions performed by the Software Programs will meet the user's requirements or will operate in the combinations that may be selected for use by the user or any third person, (c) the operation of the Software Programs will be error free in all circumstances, (d) any defect in a Software Program that is not material with respect to the functionality thereof as set forth herein will be corrected, (e) the operation of a Software Program will not be interrupted for short periods of time by reason of a defect therein or by reason of fault on the part of USDATA, or (f) the Software Programs will achieve the results desired by the user or any third person.

U.S. GOVERNMENT RESTRICTED RIGHTS. The Software is provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the government of the United States 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 in subparagraphs (c)(1) and (2) of the Commercial Computer Software—Restricted Rights clause at 48 CFR 52.227-19, as applicable. Contractor/Manufacturer is United States Data Corporation, 2435 North Central Expressway, Suite 100, Richardson, TX 75080-2722. To the extent Customer transfers Software to any federal, state or local government agency, Customer shall take all acts necessary to protect the rights of USDATA in Software, including without limitation all acts described in the regulations referenced above.

The Software Programs are furnished under a software license or other software agreement and may be used or copied only in accordance with the terms of the applicable agreement. It is against the law to copy the software on any medium except as specifically allowed in the applicable agreement. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose without the express written permission of USDATA.

Trademarks. USDATA and FactoryLink are registered trademarks of United States Data Corporation. Open Software Bus is a registered trademark licensed to United States Data Corporation.

All other brand or product names are trademarks or registered trademarks of their respective holders.

Table of Contents

Fundamentals

Preface 15

Purpose 15

Audience 15

The FactoryLink Documentation Set 15

Structure of *Fundamentals* 16

How To Use This Manual 16

Conventions 17

Getting Help 19

1 Overview 21

Development Environment 22

Run-time Environment 23

Real-time Database 24

Generating Data in Real-time Database 25

Reading Data from the Real-time Database 26

Data Logging Overview 27

FactoryLink Modules 29

General Modules 29

Modules for Accessing Stored Data 30

Modules for Logging and Retrieving Data 30

Modules for Communicating Across the Network 30

Two Domains 31

Domain Structure 32

Domains for Run-time Tasks 33

Multiple-user Environments 34

Multiple Shared Applications 34

Multiple Separate Applications 34

FactoryLink Directory Organization 35

FactoryLink Application Directory Subdirectories 35

FactoryLink Application Directory Files 36

FactoryLink System Directory Subdirectories 36

Table of Contents

Fundamentals

| | | |
|---|---|----|
| | Environment Variables | 38 |
| | Using Format Specifiers | 39 |
| 2 | <i>Alarm Logging Task Definition</i> | 43 |
| | Alarm Logging Methodology | 44 |
| | Establishing the Alarm Criteria | 45 |
| | Alarm States | 49 |
| | Alarm Grouping | 50 |
| | Parent/Child Relationship | 51 |
| | <i>Child Alarm Delay</i> | 52 |
| | <i>Child Recovery Delay</i> | 53 |
| | Hide Alarms | 55 |
| | <i>Global Hide Tag</i> | 55 |
| | <i>Group Hide Tags</i> | 55 |
| | <i>Individual Hide Tags</i> | 56 |
| | <i>Remote Groups</i> | 56 |
| | <i>Locally Redefined Unique Alarm IDs</i> | 56 |
| | Alarm Persistence | 56 |
| | <i>Activating Alarm Persistence</i> | 56 |
| | Alarm Distribution | 57 |
| | Alarm Logging | 58 |
| | Logbook | 60 |
| 3 | <i>Batch Recipe Task Definition</i> | 61 |
| | Principles of Operation | 62 |
| | Creating and Animating a Graphics Display | 64 |
| 4 | <i>Database Browser Task Definition</i> | 65 |
| | Principles of Operation | 66 |
| | Use of Logical Expressions | 68 |
| | Configuring Program Arguments | 71 |

| | | |
|---|--|----|
| 5 | <i>Database Logging Task Definition</i> | 73 |
| | Database Logging Methodology | 74 |
| | Grouping Data | 76 |
| | Ordering Data | 79 |
| | <i>Integer</i> | 79 |
| | <i>Time-stamping</i> | 79 |
| | Indexing | 80 |
| | Indexing for the Trend Task | 81 |
| | <i>Time-Base Chart Indexing</i> | 81 |
| | <i>Event Chart Indexing</i> | 81 |
| | <i>Event and Time-based Trend Chart Indexing</i> | 82 |
| | <i>Grouped Data Indexing</i> | 82 |
| | Deletion Methods | 83 |
| | <i>Record Rollover</i> | 83 |
| | <i>Subgroup Rollover</i> | 84 |
| | <i>Deleting Group Data</i> | 85 |
| | Database Logging Methods | 86 |
| | <i>Nongrouped/Nonsequenced Method</i> | 86 |
| | <i>Nongrouped/Sequenced Method</i> | 86 |
| | <i>Grouped method</i> | 86 |
| | Configuring Program Arguments | 88 |
| 6 | <i>Data-Point Logging Task Definition</i> | 91 |
| | Data-Point Logging Function | 92 |
| | <i>Preconfigured Data-Point Logging Tables</i> | 92 |
| | <i>Preconfigured Data-Point Logging Table Schemas</i> | 93 |
| | <i>Data Logged</i> | 93 |
| | Logging Methods | 94 |
| | <i>Logging Based on a Change in the Tag Value (Exception Logging)</i> | 94 |
| | <i>Logging Based on a Fixed-Time Interval</i> | 95 |
| | <i>Logging Based on a Change in a Trigger Tag</i> | 95 |
| | Logging Data | 95 |
| | <i>Logging Data During Configuration</i> | 96 |
| | <i>Logging Data Dynamically</i> | 96 |

Table of Contents

Fundamentals

| | | |
|----|--|-----|
| 7 | <i>Dynamic Data Exchange (DDE) Task Definition</i> | 97 |
| | Principles of Operation | 98 |
| | DDE Conversations | 99 |
| | <i>DDE Messages</i> | 101 |
| | Setting Up DDE Server | 102 |
| | Establishing a DDE Link | 103 |
| | <i>Using a DDE Formula</i> | 104 |
| | <i>Using a Create Link Option</i> | 104 |
| | Modifying a Linked Element | 105 |
| | DDE Client Functions | 106 |
| | <i>Read Operations</i> | 106 |
| | <i>Write Operations</i> | 107 |
| | Setting Up DDE Client | 108 |
| 8 | <i>Event and Interval Timer Task Definition</i> | 109 |
| | Principles of Operation | 110 |
| | Changing the Operating System Date and Time | 112 |
| 9 | <i>File Manager Task Definition</i> | 113 |
| 10 | <i>Historian Task Definition</i> | 115 |
| | <i>Historian Methodology</i> | 115 |
| | <i>Supported Relational Databases</i> | 116 |
| 11 | <i>Local Area Networking Task Definition</i> | 117 |
| | FLLAN Methodology | 118 |
| | Sending and Receiving Data | 119 |
| | <i>Sending Values to Remote Stations</i> | 119 |
| | <i>Receiving Values from Remote Stations</i> | 120 |
| | Local and Remote Stations | 121 |

| | | |
|----|--|-----|
| | Network Groups | 122 |
| | Using Multiple Platforms on a Network | 122 |
| | <i>Supported Protocols</i> | 122 |
| | <i>Supported Network Operating Systems</i> | 122 |
| | Monitoring the Network | 123 |
| | Local Station's Default Values | 124 |
| | <i>TX (Transmit Timeout)</i> | 124 |
| | <i>RX (Receive Timeout)</i> | 124 |
| | <i>INIT</i> | 125 |
| | <i>CALL</i> | 125 |
| | <i>MAXLEN</i> | 126 |
| | <i>BUFSIZE</i> | 127 |
| | <i>MAXSESS</i> | 127 |
| | <i>ACK</i> | 127 |
| | <i>ST (Send Timeout)</i> | 128 |
| | <i>SD (Send Delay)</i> | 128 |
| | Ordering Tag Names | 129 |
| 12 | <i>Math & Logic Task Definition</i> | 131 |
| | Uses | 132 |
| | Procedures | 132 |
| | Creating Programs | 133 |
| | <i>Configuration Tables</i> | 133 |
| | Modes | 135 |
| | <i>Interpreted Mode</i> | 135 |
| | <i>Switching from IML to CML</i> | 136 |
| | <i>Compiled Mode</i> | 136 |
| | <i>CML Operation</i> | 137 |
| | Triggering/Calling | 139 |
| | <i>Triggering</i> | 139 |
| | <i>Calling</i> | 139 |
| 13 | <i>OLE Automation Server Task Definition</i> | 141 |
| | Principles of Operation | 142 |

Table of Contents

Fundamentals

| | | |
|----|--|-----|
| 14 | <i>Persistence Task Definition</i> | 143 |
| | Persistence Overview | 143 |
| | Principles of Operation | 145 |
| | Resolving Configuration Changes | 147 |
| 15 | <i>PowerNet Task Definition</i> | 149 |
| | Using PowerNet with the FLDEMO Application | 150 |
| | Definitions | 151 |
| | PowerNet Methodology | 152 |
| | Method of Data Transfer | 155 |
| | <i>Startup</i> | 155 |
| | <i>Server-to-Client Data Transfer</i> | 155 |
| | <i>Client-to-Server Data Transfer</i> | 156 |
| | Defining Tags | 157 |
| | Tag Types | 157 |
| | Tag Naming Guidelines | 158 |
| | <i>System Added Extensions</i> | 158 |
| | Tag Type Conversion | 159 |
| 16 | <i>Print Spooler Task Definition</i> | 161 |
| 17 | <i>Programmable Counters Task Definition</i> | 163 |
| | Principles of Operation | 164 |
| | <i>Elements</i> | 164 |
| | <i>Digital and Analog Values</i> | 164 |
| | <i>Example One</i> | 165 |
| | <i>Example Two</i> | 165 |
| 18 | <i>Report Generator Task Definition</i> | 167 |
| | Reporting Methodology | 168 |

| | | |
|----|---|-----|
| | Components of a Format File | 170 |
| | <i>Keywords</i> | 170 |
| | <i>Sections of Format File</i> | 170 |
| | Placement of Reported Data | 172 |
| | <i>Location of Object Name</i> | 172 |
| | <i>Format Specifiers</i> | 172 |
| | Trigger Actions | 173 |
| | Report Format Variations | 176 |
| | Complete Triggers | 177 |
| | Escape Sequences | 178 |
| 19 | <i>Scaling and Deadbanding Task Definition</i> | 179 |
| | Principles of Operation | 180 |
| | Defining Tags | 181 |
| | Tag Naming Guidelines | 182 |
| | <i>System Added Extensions</i> | 182 |
| 20 | <i>Schema Task Definition</i> | 183 |
| | <i>Database Logging</i> | 183 |
| | <i>Data-Point Logging</i> | 184 |
| | <i>Database vs. Data-Point Logging</i> | 185 |
| | <i>Trending</i> | 186 |
| | <i>Database Logging Schemas</i> | 186 |
| 21 | <i>Trending Task Definition</i> | 189 |
| | Trending Methodology | 190 |
| | <i>Real-time Only Trending</i> | 190 |
| | <i>Historical Trending</i> | 191 |
| | Chart Types | 193 |
| | <i>Time-based Chart</i> | 193 |
| | <i>Event Chart</i> | 194 |
| | Switching Between Real-time and Historical Mode | 197 |
| | Zooming and Panning | 197 |
| | Pen Types | 197 |
| | Value Cursor | 198 |
| | Configuring Program Arguments | 199 |

Table of Contents

Fundamentals

| | | |
|----|---|-----|
| 22 | <i>Defining Tags</i> | 201 |
| | Tag Naming Guidelines | 202 |
| | <i>Defining Tags While in the Configuration Manager</i> | 203 |
| | <i>Defining Tags While in the Application Editor</i> | 205 |
| | <i>Editing a Tag</i> | 206 |
| | Defining Element Arrays | 207 |
| | <i>Defining One-dimensional Arrays</i> | 207 |
| | <i>Defining Multi-dimensional Arrays</i> | 208 |
| | <i>Maximum Number of Arrays</i> | 209 |
| | Using Tags as Triggers | 210 |
| | Predefined Elements | 210 |
| 23 | <i>Using the Configuration Manager</i> | 211 |
| | Opening the Configuration Manager | 212 |
| | <i>Windows NT or Windows 95 Platform</i> | 212 |
| | <i>OS/2 Platform</i> | 213 |
| | <i>UNIX Platform</i> | 213 |
| | Configuration Manager Display | 215 |
| | Working with Text-entry Panels | 216 |
| | Working with Structured Configuration Panels | 217 |
| | Scrolling a Window or Panel | 219 |
| | <i>Vertical Scroll Bar</i> | 219 |
| | <i>Horizontal Scroll Bar</i> | 220 |
| | Working with Multiple Development Applications | 221 |
| | <i>Opening a New Development Application</i> | 221 |
| | <i>Sharing Information Between Applications</i> | 222 |
| | Opening Multiple Configuration Tables | 225 |
| | Working with Tags | 226 |
| | <i>Viewing the Number of Tags Defined</i> | 226 |
| | <i>Viewing a List of Real-time Database Elements</i> | 227 |
| | <i>Viewing Element Cross Reference List</i> | 229 |

| | | |
|----|--|-----|
| | <i>Searching for a Tag</i> | 231 |
| | <i>Deleting a Tag Definition</i> | 233 |
| | <i>Changing a Tag Definition</i> | 235 |
| | Viewing Miscellaneous Information | 236 |
| | <i>Viewing Domain List</i> | 236 |
| | <i>Viewing the Current Version of FactoryLink</i> | 238 |
| | <i>Viewing Field Choices</i> | 239 |
| | Editing Configuration Panels | 240 |
| | <i>Inserting a Blank Line</i> | 240 |
| | <i>Deleting a Line Entry</i> | 240 |
| | <i>Copying Contents of a Line Entry</i> | 241 |
| | <i>Cutting a Line Entry</i> | 241 |
| | <i>Pasting a Copied or Cut Line Entry</i> | 242 |
| | <i>Searching for a Text String</i> | 242 |
| | Getting Help | 243 |
| | Managing Text-entry Panel Files | 244 |
| | <i>Saving File Contents</i> | 244 |
| | <i>Merging a File</i> | 244 |
| | <i>Checking Syntax</i> | 245 |
| | <i>Clearing Errors</i> | 245 |
| | <i>Changing Font Size</i> | 245 |
| | Reporting | 246 |
| | Configuration Manager Messages | 248 |
| 24 | <i>Using System Configuration</i> | 273 |
| | Editing Task Information | 281 |
| | <i>Viewing Domain Associations</i> | 281 |
| | <i>Adding New Tasks</i> | 281 |
| | Changing Domain Associations | 283 |
| | Setting the Run-Time Graphics Windows | 284 |
| | Changing the Windows Color Scheme | 285 |
| | Archiving Error Messages | 286 |
| | Resizing and Moving Screen Components | 289 |
| | Calculating the Number of FactoryLink Processes | 290 |

Table of Contents

Fundamentals

| | | |
|----|--|-----|
| 25 | <i>Using Run-Time Manager</i> | 293 |
| | Setting Up Program Arguments | 293 |
| | <i>Tuning Kernel Memory</i> | 293 |
| | Setting Command Line Options | 295 |
| | Accessing Run-Time Manager | 299 |
| | <i>On Windows NT and Windows 95 Platforms</i> | 299 |
| | <i>On OS/2 Platform</i> | 299 |
| | <i>On UNIX Platform</i> | 300 |
| | Run-Time Manager User-Interface Screen | 304 |
| | <i>Task Controls</i> | 305 |
| | <i>Application Controls</i> | 306 |
| | <i>Display Controls</i> | 307 |
| | Run-time Manager Messages | 308 |
| | <i>Text Messages</i> | 308 |
| | <i>Messages with Error Numbers</i> | 314 |
| | <i>Correcting Internal Errors</i> | 320 |
| 26 | <i>Using Run-Time Monitor</i> | 321 |
| | Accessing the Run-Time Monitor | 322 |
| | Viewing the Current Value of Elements | 324 |
| | <i>Adding Elements to a Watch List</i> | 325 |
| | <i>Deleting Elements from a Watch List</i> | 326 |
| | <i>Finding Elements in a Watch List</i> | 326 |
| | <i>Storing Watch List</i> | 327 |
| | <i>Retrieving Watch List</i> | 328 |
| | Reading and Writing a Real-Time Database Element | 329 |
| | Monitoring FactoryLink Processes | 331 |
| | Using Run-Time Monitor Commands | 335 |
| | <i>Terminating a Task</i> | 336 |
| | <i>Exiting Commands Window</i> | 336 |
| | <i>Accessing Help</i> | 336 |

| | | |
|----|---|-----|
| | <i>Reading an Element</i> | 337 |
| | <i>Writing to an Element</i> | 339 |
| | <i>Working with Batch Files</i> | 341 |
| | Monitoring the Status of the Real-Time Database | 343 |
| | Terminating All FactoryLink Tasks | 346 |
| | Exiting RTMON | 347 |
| | Run-time Manager Messages | 348 |
| | <i>Text Messages</i> | 348 |
| | <i>Messages with Error Numbers</i> | 354 |
| | <i>Correcting Internal Errors</i> | 360 |
| 27 | <i>FactoryLink Utilities</i> | 361 |
| | FLNEW | 362 |
| | FLREST | 392 |
| | FLSETLNG | 395 |
| | FLTEST and FLDEMO | 396 |
| | <i>On Windows NT and Windows 95 Operating Systems</i> | 396 |
| | <i>On OS/2 Operating Systems</i> | 398 |
| | On UNIX Operating Systems | 400 |
| | <i>Restoring FLDEMO</i> | 400 |
| | <i>Restoring FLTEST</i> | 400 |
| | Module Dependencies | 401 |
| | FLCONV | 402 |
| | FLSAVE | 403 |
| | CTGEN | 406 |
| | CDBLIST | 406 |
| | CTLIST | 406 |
| | DBCHK | 407 |
| | EXPLODE | 408 |
| | keyinst | 409 |
| | flkeyval | 409 |
| | FLSHM | 410 |
| | UKEY | 411 |
| | Utility Messages | 412 |
| | <i>EXPLODE</i> | 414 |
| | <i>FLSAVE</i> | 415 |
| | <i>FLREST</i> | 416 |

Table of Contents

Fundamentals

28 *FactoryLink Lite* 419

 Defining Tags 420

 Tag Naming Guidelines 421

System Added Extensions 421

 Developer-Defined Tag Element Maximum 422

I/O Element Maximum 422

 Configuration Guidelines 424

 FactoryLink Lite Error Messages 426

29 *Glossary* 429

Index 459

Preface

.....

PURPOSE

FactoryLink Fundamentals presents basic concepts necessary to understand how FactoryLink works.

This guide is prepared to present the technical information programmers of this system need to complete their applications.

The following guidelines help focus our purpose and goals to meet customer requirements:

- Accuracy is paramount—This FactoryLink documentation must provide accurate and reliable information and procedures.
- Time is valuable—This FactoryLink documentation must guide the programmer through what he needs to know quickly and efficiently.

AUDIENCE

The major audience of this manual is programmers who design and construct programs to configure tasks that facilitate functions performed by standard FactoryLink tasks.

In addition, FactoryLink Customer Support Services personnel use the procedures and examples included here to help you develop and troubleshoot your applications.

THE FACTORYLINK DOCUMENTATION SET

The FactoryLink Documentation Set is divided in a Complete set of manuals and a Basic set of manuals. Included in the Complete set are

- Installation Guide
- Release Notes
- Tutorial
- Fundamentals
- Application Editor
- Device Interface Guide
- Reference Guide
- Configuration Guide
- Power SPC Configuration Guide

- **PREFACE**
- *Structure of Fundamentals*
-
-

- PowerVB Language Reference Guide
- Programmer's Access Kit
- Comprehensive Index
- Product Matrix

The Basic set of manuals includes only the first seven manuals listed along with a Master Index that includes entries for Fundamentals, Application Editor, and the Reference Guide only.

STRUCTURE OF *FUNDAMENTALS*

Fundamentals is part of the Complete and Basic sets of manuals in the overall FactoryLink Documentation Set.

This manual is divided into twenty-eight chapters plus a Glossary.

Chapters 1 - 21—General overview of the FactoryLink product and overviews of each individual task used to configure a FactoryLink application.

Chapter 22—Working with Tags in the Configuration Manager

Chapter 23—Using the Configuration Manager

Chapter 24—Using System Configuration

Chapter 25—Using Run-Time Manager

Chapter 26—Using Run-Time Monitor

Chapter 27—FactoryLink Utilities

Chapter 28—FactoryLink Lite

Glossary

HOW TO USE THIS MANUAL

The material in this manual is presented in a learning order. We recommend you read the entire manual to familiarize yourself with all the information before you proceed to develop your application.

CONVENTIONS

The material in the Documentation Set adheres to the guidelines published in *The Digital Technical Documentation Handbook* by Schultz, Darrow, Kavanagh, and Morse; *Developing International User Information* by Jones, Kennelly, Mueller, Sweezy, Thomas, and Velez; and corporate style guidelines.

This manual uses the following conventions.

| Convention | Description |
|----------------------------|---|
| ... | Horizontal ellipsis points indicate the omission of material from an example. The information is omitted because it is not important to the topic being discussed. |
| | Vertical ellipsis points indicate the omission of information from an example or command format. The information is omitted because it is not important to the topic being discussed. |
| <i>italic type</i> | Italic type is used to denote user-supplied variables in command examples. Italic type also sets off references to specific documents. |
| monospace type | Monospace type is used to denote command names and code examples or example output. |
| bold monospace type | Bold monospace type is used in command examples to indicate words that must be typed literally. |
| sans serif type | Sans Serif type is used to set off field names, button names, and keys on the keyboard. |
| blue type | Blue type is used for headings and to call attention to information within the text. |
| press nnnnn | Press is used to denote a key on the keyboard. The key name will appear in a sans serif type. |
| click on nnnnn | Click on is used to denote a button on the screen. The button name will appear in a sans serif type. |

- **PREFACE**
- *Conventions*
-
-

| Convention | Description |
|------------------------|--|
| Shift+F1 | <p>The + indicates the keys must be pressed simultaneously.</p> <p>Shift+F1 indicates you hold down the Shift key while you press another key or mouse button (indicated here by F1).</p> <p>Other key combinations are presented in the same manner.</p> |
| F1 F2 F3 | <p>The space between the key callouts indicates press and release.</p> <p>The key sequence F1 F2 F3 indicates you press and release F1, then F2, and then F3.</p> <p>Other key combinations are presented in the same manner.</p> |
| File>Open | <p>The > indicates a progression through a menu sequence.</p> <p>File>Open indicates you choose Open from the File menu to perform the required action.</p> <p>Other menu sequences are presented in the same manner.</p> |
| FLAPP\user\drw\mydrw.g | <p>The \ indicates the directory structure for the listed file.</p> <p>FLAPP\user\drw\mydrw.g indicates the drawing file mydrw.g is located in the drw sub-directory of the user sub-directory to the FLAPP directory.</p> <p>Other directory structures are presented in the same manner.</p> |
| [] | <p>Brackets indicate an optional argument. You can choose none, one, or all of the options.</p> |
| { } and | <p>Braces indicate a choice. You must choose one of the elements. The vertical bar separates choices within braces.</p> |

Example Syntax

Example syntax using these conventions is provided below:

command *input_file* [*input_file...*] {*a/b*} *output_file*

where

command is typed as it is displayed in the syntax.

input_file indicates a variable the user supplies.

[*input_file...*] indicates the user can optionally supply multiple input file names, each name separated by a space.

{*a/b*} indicates either the a or b must be specified as an argument.

output_file indicates the user must specify an output file.

GETTING HELP

Contact your Sales or Customer Support Representative for help with troubleshooting problems.

Also, help files are included for each configuration panel. Click on Help on the panel menu bar to access these files.

- **PREFACE**
- *Getting Help*
-
-

Overview

FactoryLink is a software development tool designed to build a modular, multi-tasking application that automates and controls a process, such as the production of goods at a factory, the movement of liquid or gas down a pipeline, or the periodic collection of data. You use the unlimited capabilities of FactoryLink to build an application that performs the task you want to automate for your processes.

FactoryLink has two operating environments.

- **Development**—The activities required to automate and control a process are configured in the development environment creating a practical application. You configure an application using the Configuration Manager Main Menu. Refer to “Using the Configuration Manager” on page 211 for details on starting and using the Main Menu.
- **Run time**—When the application created in the development environment is started, it executes in the run-time environment. You control and monitor the status of the application using the Run-Time Manager, which allows you to start, monitor, and stop individual run-time tasks. Refer to “Using Run-Time Manager” on page 293 for details on starting an application and using the Run-Time Manager.

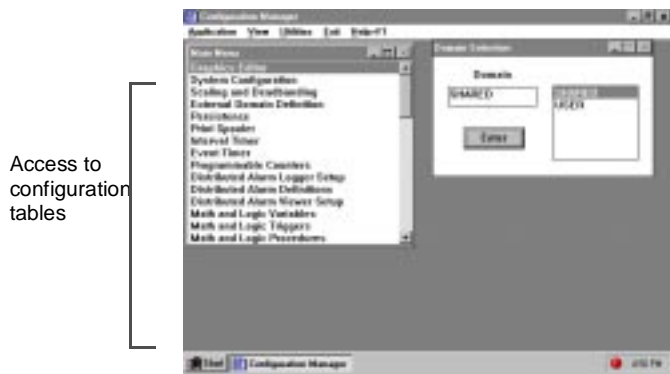
FactoryLink is a set of programs; each program performs a specific activity in the automation process, such as the collection and storage of data, the generation of reports, or the management of files. These programs are called modules in the development environment and tasks in the run-time environment.

This chapter describes the concepts necessary to understand how FactoryLink works. It also describes each module that can be configured to automate your process.

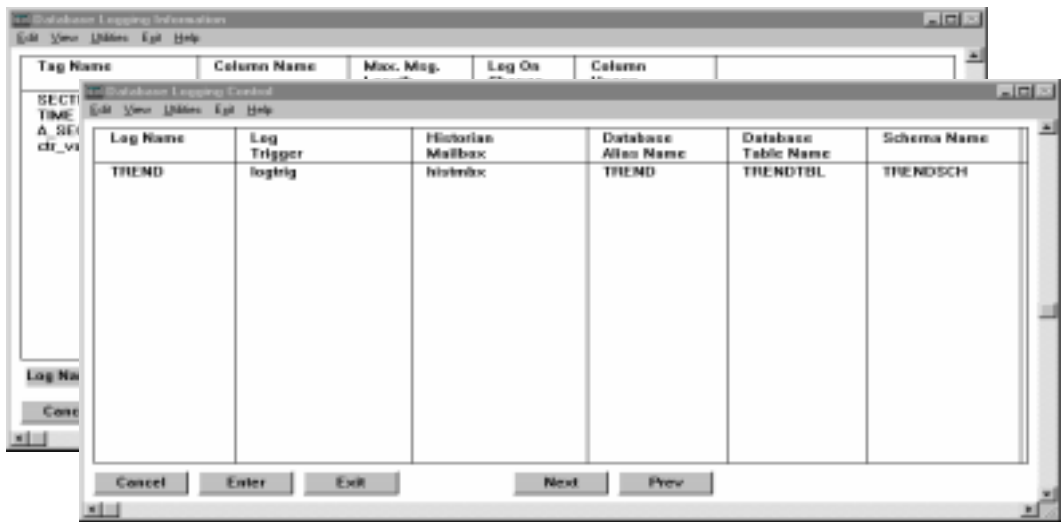
- OVERVIEW
- Development Environment
-
-

DEVELOPMENT ENVIRONMENT

A FactoryLink application is configured in the development environment. This environment contains a series of configuration tables for each module. Refer to the “FactoryLink Modules” on page 29 for a list of modules. You open these tables by choosing a specific menu option from the Main Menu.



Each option is comprised of one or more panels. For example, the Logger table is comprised of two panels: the Database Logging Control panel and the Database Logging Information panel.



You must fill out these panels to configure the module. Refer to the *FactoryLink Configuration Guide* for details on configuring each module.

The information entered in these panels is stored in dBASE IV compatible files in the directory where the FactoryLink application files are stored. Refer to “FactoryLink Directory Organization” on page 35 for details on where these files reside. These files are identified by their .cdb extension and their corresponding index files are identified by their .mdx extension.

RUN-TIME ENVIRONMENT

The run-time environment uses the information provided in the configuration tables to build an application. When the FactoryLink application is started, the contents of the dBASE IV files are translated and stored in binary files containing a .ct extension. Where these files are located depends on the domain associated with the data. Refer to “FactoryLink Directory Organization” on page 35 for details. The name assigned to each file uses the following form:

`module_name.ct`

where

`module_name` the name of the task as it is displayed in the system configuration task.

- **OVERVIEW**
- *Real-time Database*
-
-

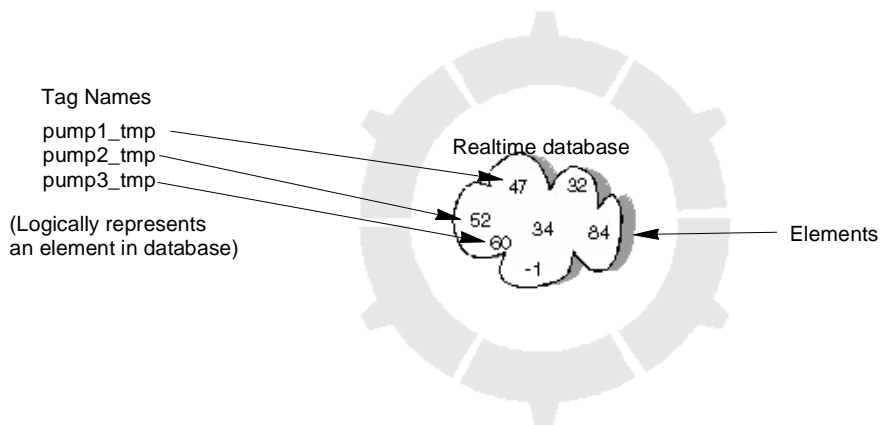
REAL-TIME DATABASE

A Real-Time Database is central to all FactoryLink tasks in real-time mode. This Real-Time Database resides in memory and acts as both a storage device and an interprocess communication mechanism.



All FactoryLink tasks share information in the Real-Time Database by reading from or writing to the Real-Time Database. Data is never passed directly between two tasks.

The Real-Time Database contains all the data values to be shared between the application tasks. When configuring the FactoryLink application, the developer defines which values are to be held within the Real-Time Database. These data values are called tags. Each tag has a unique name and a unique location within the Real-Time Database.



Refer to “Defining Tags” on page 201 for details on assigning tag names and for a list of predefined elements shipped with FactoryLink.

GENERATING DATA IN REAL-TIME DATABASE

The database stores data that has been:

- Collected from a remote device



- Computed by a FactoryLink task

$$CBAR = \frac{\sum c}{k}$$

- Manually entered by a user



When data is collected and stored in the database, other tasks can open and manipulate it. A task can write information to an element in the Real-Time Database using one of two types of writes: a normal write or a forced write.

Normal writes only write to an element in the Real-Time Database if the new value is different from the existing value. In this way, system resources are not used unnecessarily.

Forced writes write to an element whether or not it has changed. Use this type of write if you want to set the change-status flag even if the data has not changed. Refer to “Reading Data from the Real-time Database” on page 26 for a discussion on the change-status flag.

- **OVERVIEW**
- *Reading Data from the Real-time Database*
-
-

READING DATA FROM THE REAL-TIME DATABASE

A task gets the value of an element in the Real-Time Database using a read operation. Once the task has this value, it can perform functions on it, such as displaying on a graphical user-interface, transmitting to an external device, or sending it to a relational database for archiving.

Read operations can be triggered by an event (such as when a product passes an electronic eye) or can occur only if the data element changes. A read operation that occurs only when the data changes is referred to as exception processing. Because large blocks of data can be transferred between tasks, and because only the changed values are processed, exception processing significantly optimizes performance.

Exception processing is possible because of the structure of the FactoryLink Real-Time Database elements. A Real-Time Database element consists of a number of bits, one or more of which contain the element's value. Thirty-one of these bits are reserved to function as change-status flags. Each FactoryLink task is assigned one of these bits which it uses to determine the element's change status.

The change-status flag is either set to 1 (ON) or 0(OFF). One (1) indicates to the task the value of the element has changed since the last time the task read the element. Zero (0) indicates to the task the value of the element has not changed. When an element in the database is written, all the bits are automatically set by FactoryLink.

DATA LOGGING OVERVIEW

As data is collected or computed by FactoryLink, it is stored as a data element with a unique tag name in a Real-Time Database. Each time data for a tag is collected or computed, the new value is stored in the Real-Time Database, overwriting the old data. If you want to retain this data so that it can be reviewed, compared and/or analyzed, you can log it to a historical database.

The major aspects of the data logging function in FactoryLink are:

- The Schema
- The Historian
- The Logger

Schema: In FactoryLink, the relational databases are configured in a table format consisting of rows and columns. The structure of a table (the number, size and content of the rows and columns) is defined by its schema. When configuring a database you must specify what elements for which to collect values and where to place the values: what database, table, row, and column. Refer to “Schema Task Definition” on page 183 for more information about schemas.

Historian: The historian serves as an interface between the FactoryLink Real-Time Database and the external relational database to which information is being sent or retrieved. Refer to “Historian Task Definition” on page 115 for more information about historians.

Logger: FactoryLink has two different logging tasks — Data-Point and Database. Both tasks read data from the Real-Time Database and send it to the disk-based relational database via a FactoryLink Historian.

- Data-Point Logging was developed to simplify the task of logging data by providing preconfigured tables. Multiple shared numeric-value tags can be stored in the same database and sorted later if necessary. It also allows you to add or remove tags from the list of tags being logged during run time. Data-Point Logging builds a database that will capture, for the specified tag:

Date/time

Tag name

Tag value

Data-Point Logging is best suited for situations in which the tags that are to be logged do not need to be grouped together. Refer to “Data-Point Logging Task Definition” on page 91 for more information on Data-Point Logging.

- **OVERVIEW**
- *Data Logging Overview*
-
-

- **Database Logging** task allows you to create a table and specify which tags to capture in that table. When the value of any tag changes, the value of all tags in the table is logged. Database Logging provides the ability to group tags in a database table. In addition, event-based data can be logged using a sequence key rather than a time key. Refer to “Database Logging Task Definition” on page 73 for more information on Database Logging.

Data-Point vs. Database Logging

Prior to configuring the logging function you should determine which logging task you will be using for a particular situation.

Data-Point Logging is best for situations when you want to:

- Log a tag only when its value changes
- Use preconfigured tables and eliminate the time spent setting up tables
- Be able to index on log time or tag name or both
- Sort all logs of a tag in order of occurrence
- Configure a tag to be a dynamic pen on a trend chart
- Dynamically change the list of tags being logged during run time.

Database Logging is best for situations when you want to:

- Log all tags when any tag changes or is triggered
- Simultaneously log a group of like (logically associated) tags
- Group data logged based on some criteria
- Configure a table column to be a dynamic pen on a trend chart.

For example, if you had a tank in which you were monitoring the process temperature and had a single probe in the tank, you would probably want to use Data-Point Logging to track the tag value. If you had six probes in the same tank and you wanted to see the value returned from each probe at a given point in time, you would probably want to use the Database Logging task.

FACTORYLINK MODULES

FactoryLink is comprised of modules with unique functions.

General Modules

- **Application Editor**—Draws and animates graphical user-interfaces to the FactoryLink application.
- **Batch Recipe**—Stores recipes on disk for manufacturing a product. These recipes can be sent to an external device at a given time to control the product manufactured.
- **Device Interface**—Communicates bi-directionally between the Real-Time Database and one or more external devices, such as programmable logic controllers (PLCs) and remote terminal units (RTUs).
- **DXF Files**—Convert original AutoCAD drawings into .gx files (an ASCII text-based file) that FactoryLink can import.
- **Dynamic Data Exchange**—Updates data automatically and regulates how and when data is passed between applications.
- **Event and Interval Timer**—Works in synchronization with the system clock. It updates global information FactoryLink uses as the current time, the day of the week, and the month.
- **Persistence**—Saves the values of an active FactoryLink application at predetermined times so if FactoryLink shuts down unexpectedly, useful data is not lost.
- **Power VB**—Allows Microsoft Visual Basic[®] compatible scripts to be attached to FactoryLink graphic objects.
- **Print Spooler**—Permits you to direct data to printers or other devices with parallel interfaces and also to disk files.
- **Programmable Counter**—Provides count-per-unit-of-time measurements and provides event delays, such as defining a trigger to unlock a door and then specifying a delay before the door locks again.
- **Scaling and Deadbanding**—Converts or scales incoming raw data to a different value range and indicates a dead or non-recalculating band around a scaled value.
- **Utilities**—Provides ways to do general maintenance of your application.

- **OVERVIEW**
- *FactoryLink Modules*
-
-

Modules for Accessing Stored Data

- **Math & Logic**—Performs math on data collected in the Real-Time Database. The results can be used for writing new data elements, inclusion in reports, or triggering events.
- **Report Generator**—Generates reports using data from the Real-Time Database. These reports can be archived to disk or sent to a printer.

Modules for Logging and Retrieving Data

- **Database Browser**—Retrieves, updates, deletes, and inserts data into an existing database.
- **Database Logger**—Copies data from the memory-based Real-Time Database to historical disk-based relational databases.
- **Data-Point Logger**—Simplifies the task of logging data by providing preconfigured tables.
- **Distributed Alarm Logger**—Checks real-time data for permitted limits, generates alarms if limits are exceeded, and copies the alarms to historical disk-based relational database.
- **Historian**—Acts as a conduit for information between FactoryLink tasks that send data to or retrieve data from a relational database.
- **Schemas**—Define table structures, such as row and column number, size, and content.
- **Statistical Process Control**—Displays real-time or historical data on statistical charts.
- **Trending**—Displays real-time or historical data on trend charts.

Modules for Communicating Across the Network

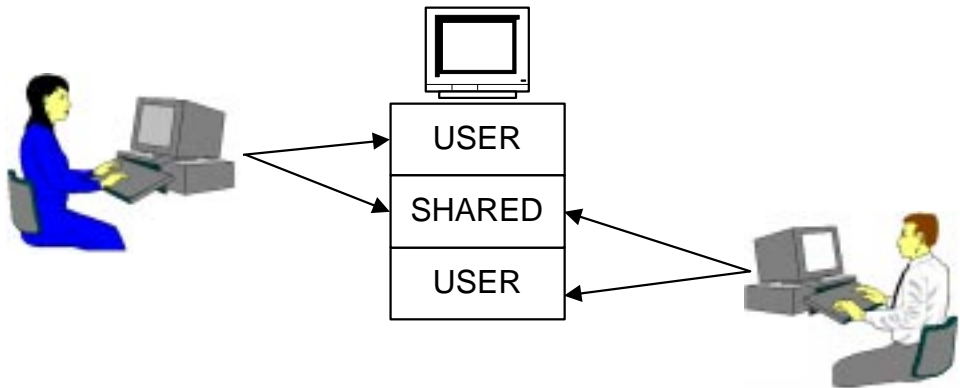
- **FactoryLink Local Area Network**—Links networked applications providing transfer of real-time data elements from one station to the next.
- **FLOCX**—Is an ActiveX control designed to interface with the OLE Server task.
- **File Manager**—Manages files on local drives or remote servers and transfers files from one station to the next.
- **PowerNet**—Shares real-time data between applications across a network.

Details on how to configure each module can be found in the *FactoryLink Configuration Guide*.

TWO DOMAINS

There are two run-time environments called domains. These are the SHARED domain and the USER domain. During configuration, you associate tasks, tables, and elements with a specific domain so you can configure a FactoryLink application to suit your needs.

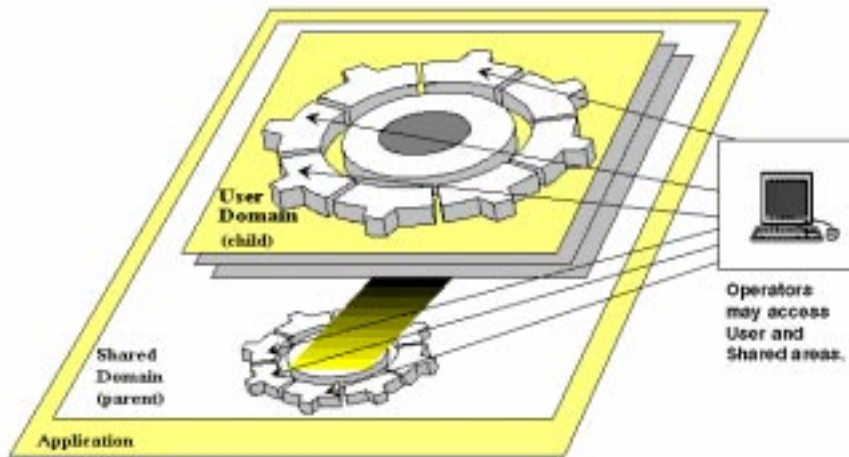
- You can give a group of users independent access to the same FactoryLink run-time tasks such as Graphics, Math & Logic, and Statistical Process Control. This lets users access the same data at the same time with two different tasks.
- You can create an application that lets multiple users of a single run-time system use the tasks independently without the users sharing data; that is, users can simultaneously run the same tasks, but each user's data is unique. For example, one user can employ the Statistical Process Control task to evaluate the consistency of an assembly sequence while another uses it to report anomalies in a packaging process elsewhere in the factory.
- You can set up multiple FactoryLink applications to run on one operating system. For example, applications for development, testing, and production, can all run on one machine.



- **OVERVIEW**
- *Two Domains*
-
-

Domain Structure

Domains exist in a parent/child hierarchy. The following illustrates the shared domain is like a parent upon which a child domain (user) is based.



A SHARED domain has the following characteristics:

- **Accessibility**—All operators within a user domain can access data existing in a shared domain.
- **Frequent use**—Frequently used items need only be defined once in the shared area. This characteristic eliminates duplication of effort and reduces configuration time.
- **Automatic data changes**—Data changes are passed immediately to all users who are accessing the modified data in a user domain.
- **Global data processing**—Processing activities, such as computations, data accessing, and data logging, are performed once and then are available to all users accessing the shared area.

A USER domain has the following characteristics:

- **Private copies of data**—You can configure a single application and then specify the number of instances (copies) of that domain to be executed at run time. The number of instances allowed is the number of users that can simultaneously interact with the run-time system.
- **Task Independence**—Each user has access to the same FactoryLink run-time tasks such as Graphics, Math & Logic, and Statistical Process Control but is able to run these tasks independently from the tasks other users are running. Users can share data to perform different real-time tasks within the same application.

Domains for Run-time Tasks

Run-time tasks are associated with a domain. The domain controls whether or not the run-time task can share data and which task uses data unique to a user. Choosing the appropriate domain for a task in a multi-user application is critical. The defaults that provide the best performance for most applications are listed in the following table.

Table 83-1 Optimal Domain Selection

| Shared Domain | User Domain | Both Domains |
|---------------------------|--------------|--------------|
| Timer | Graphics | Run Manager |
| Report Generator | File Manager | Math & Logic |
| Batch Recipe | Trending | Counter |
| File Manager Server | Browser | Persistence |
| External Device Interface | SPC View | RTMON |
| LAN | SPR | DALOG |
| Historian | | |
| Logger | | |
| SPC Logger | | |
| Print Spooler | | |

- **OVERVIEW**
- *Multiple-user Environments*
-
-

During application planning, review these domain defaults to verify compatibility with a specific application's needs. When an application has special requirements, you can run the task in a different domain, except for the EDI task which must be run in the shared domain.

In addition to these domain defaults, the Main Menu has a domain selection feature you must set before configuring each task. If the domain chosen during configuration does not match the default set for the domain at run time, the application will not run as intended. You must configure trending in the user domain if it is going to run in the user domain.

MULTIPLE-USER ENVIRONMENTS

The UNIX operating system supports multiple users. You can configure use by multiple users in one of two ways:

- Multiple users can share the same application
- Multiple users can access separate applications.

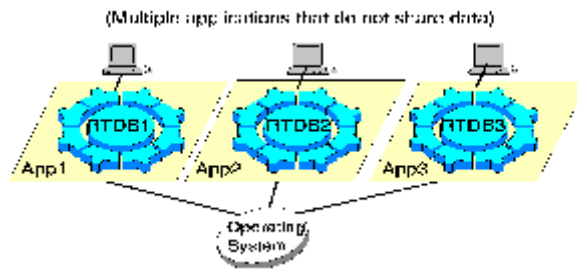
FactoryLink's platform independence lets you create applications on single-user computer systems, such as Windows and OS/2, and port them to run on multi-user systems.

Multiple Shared Applications

In a multi-user application each user executes in their own user domain; each user domain has its own copy of user data but shares data from the shared domain.

Multiple Separate Applications

In a multi-user application, each application has its own Real-Time Database and does not share data between them.



FACTORYLINK DIRECTORY ORGANIZATION

All FactoryLink files are organized into two main directories.

- FactoryLink application directory—Contains all FactoryLink application files.
- FactoryLink system directory—Contains all FactoryLink program files.

FactoryLink Application Directory Subdirectories

The FactoryLink application directory contains FactoryLink application files. The default name of this directory depends on your operating system platform.

- On UNIX platforms, the default directory is /usr/flapp.
- On all other platforms, the default directory is \FLAPP.

The application files are organized into the following subdirectories. Any subdirectories found in the application directory not listed here are reserved for future use.

- | | |
|--------|--|
| ASC | ASCII configuration database files that store information about the elements. The files in this subdirectory contain a .asc extension. They are used to import/export configuration data from one application directory to another, typically from one platform to another during multi-platform saves and restores. |
| CT | Binary files produced at run time containing non-domain specific data extracted from the database tables. The files in this subdirectory contain a .ct extension. Domain-specific ct files exist in the FactoryLink domain subdirectory. |
| DCT | Communication support tables. The files in this subdirectory contain a .dct extension. |
| NET | Local and group files used by the FactoryLink Local Area Networking task to determine communication parameters and remote node logical groupings. |
| RPT | Report Generator format files that define the format for reports. The files in this subdirectory contain a .fmt extension. |
| SHARED | Files specific to the shared domain. |
| USER | Files specific to the user domain. |

- **OVERVIEW**
- *FactoryLink Directory Organization*
-
-

FactoryLink Application Directory Files

The application directory also contains the following files.

| | |
|----------|--|
| FLAPP.ID | Application identification file. |
| *.cdb | Configuration database files that store information about elements, such as name, type, number of writes specified by the defining task, and number of references. |
| *.mdx | Index files used by the Main Menu in conjunction with the .cdb files. |
| *.cnp | Main Menu cut and paste work files. |
| *.exp | Main Menu import/export output files. |

FactoryLink System Directory Subdirectories

The FactoryLink system directory contains all FactoryLink program files. The default name of this directory depends on your operating system.

- On Windows NT platforms, the default directory is \FLNT.
- On Windows 95 platforms, the default directory is \FLWIN.
- On OS/2 platforms, the default directory is \FLOS2.
- On UNIX platforms, the default directory is /usr/flink.

The program files in this directory are organized under the following subdirectories.

| | |
|-------|---|
| AC | Text files that function as attribute catalogs to inform the Main Menu of the format of the configuration tables. They also control entry criteria. |
| BLANK | Files used by FactoryLink utilities that manage applications, such as flnew, flsave, flrest, and flconv. |
| BIN | Command script files and executable program files for each FactoryLink task. |
| CML | Default make file for Compiled Math & Logic task. |
| CTGEN | Configuration database conversion script files. |
| DRW | System files used by the Graphics task and by the Application Editor. |
| EDI | Subdirectory for External Device Interface protocol modules. |

| | |
|-----|--|
| INC | C-language include files for options such as Compiled Math & Logic and the Programmer's Access Kit. |
| KEY | Text files used by the Main Menu to translate text table entries into binary values to be placed in configuration tables. |
| LIB | Library files and objects. |
| MPS | Demo application, new application, and test application files. |
| MSG | Help files used with the Main Menu and Application Editor and error message files for FactoryLink tasks. |
| OPT | Files needed to control the FactoryLink options available with your application. |
| SRC | External Device Interface Programmer's Access Kit C-source files, libraries, and sample protocol module and sample Programmer's Access Kit source and makefiles. |

Windows NT and Windows 95 Platforms

The Windows NT and Windows 95 platforms contain the following subdirectories under the application directory.

| | |
|-----|---|
| BIN | FactoryLink command files (.bat extension) and executable program files for each FactoryLink task (.exe extension) and dynamic link library files (.dll extension). |
| LIB | Object, image library, and object library files (.obj extension). |

OS/2 Platform

The OS/2 platform contains the following subdirectories under the application directory.

| | |
|---------|---|
| BIN | FactoryLink command files (.cmd extension) and executable program files for each FactoryLink task (.exe extension). |
| INSTALL | Files used during FactoryLink installation. |
| LIB | Object files (.obj extension), library files (.lib extension) and dynamic link library files (.dll extension). |

- **OVERVIEW**
- *Environment Variables*
-
-

UNIX Platforms

The UNIX platform contains the following subdirectories under the application directory.

- BIN** FactoryLink command files and executable program files. Some protocol module executable files have an .exe extension.
- LIB** Object (.o extension) and library (.a extension) files.

ENVIRONMENT VARIABLES

Because multiple copies of FactoryLink can be running concurrently, the system uniquely identifies an application by a combination of its application name, domain name and user name. These names are defined as environment variables. You can set these environment variables either in a system configuration file so they are automatically supplied when you start the application or at the system prompt before starting the application. The environment variables you can set are listed in the following table.

Table 83-2 Environmental Variables

| Variable | Description |
|--------------------------|---|
| FLINK = <i>flink_dir</i> | Where <i>flink_dir</i> defines the full path, including the drive name, of the directory containing the FactoryLink program files. |
| FLAPP = <i>flapp_dir</i> | Where <i>flapp_dir</i> defines the full path, including the drive name, of the directory containing your application files. |
| FLDOMAIN = <i>domain</i> | Where <i>domain</i> defines the domain you are starting. This can either be shared or user. On single-user platforms, <i>domain</i> should be user. This starts both domains. On multi-user platforms, <i>domain</i> is the domain you want started in the window. You must specify two windows, one for each domain. |
| FLNAME = <i>app_name</i> | Where <i>app_name</i> defines the name of the application to start, which points to a Real-Time Database. |

Table 83-2 Environmental Variables

| Variable | Description |
|---------------------------|--|
| FLUSER = <i>user_name</i> | Where <i>user_name</i> defines the logical user name. |
| FLOPT = <i>opt_dir</i> | Where <i>opt_dir</i> defines the full path, including the drive name, of the directory containing the FactoryLink's license information. |

When using environment variables in path names, you can enter the name of the environment variable surrounded by braces { } and FactoryLink extends the pathname using the default setting.

USING FORMAT SPECIFIERS

Format specifiers allow you to define the format for all or part of an output string. The following FactoryLink tasks support the use of format specifiers:

- File Manager
- Batch Recipe
- Report Generator
- Alarm Supervisor

Format specifiers permit you to define a variable when a literal is expected. Variable specifiers can consist of two types of objects.

- Ordinary characters, which are copied literally to the output stream
- Format specifiers, which indicate the format in which variable information will be displayed

- **OVERVIEW**
- *Using Format Specifiers*
-
-

Format specifiers use the following form:

`% [flags][width][.prec]type`

where

| | |
|--------------------|---|
| <code>%</code> | Always precedes a format specifier. |
| <code>flags</code> | Controls the format of the output. This can be one of the following. <ul style="list-style-type: none"> - Left-justifies within the field. If you do not specify this flag, the field is right-justified. 0 Fills the spaces to the left of the value with zeros until it reaches the specified width. |
| <code>width</code> | Specifies minimum field width, in decimal. This field is not valid for floating point notations. Floating point fields are always padded with spaces. |
| <code>.prec</code> | Controls the precision of the numeric field. What precision defines depends on the format type specified by the <i>type</i> variable. For exponential (type e) or floating point (type f) notations, specify the number of digits to be printed after the decimal point. For short version of exponential or floating point notations (type g), specify the maximum number of significant digits. For all other types, specify the minimum number of digits to print. Leading 0s are added to make up the necessary width. |
| <code>type</code> | Specifies the character or numeric type for the value. This can be one of the following. d = decimal s = string ld = long decimal e = exponential notation of the following form <code>[-]m.nnnnnnnE[+-]xx</code> f = floating-point notation of the following form <code>[-]mmmm.nnnnnnn</code> g= use shorter of e or f u = unsigned decimal |

- o = unsigned octal
- x = unsigned hexadecimal using a - f
- X = unsigned hexadecimal using A - F

Examples of valid format specifiers for each FactoryLink data type are displayed in the following table. For additional information about format specifiers, see any ANSI-C reference manual.

Table 83-3 :Format Specifiers

| Type of Element | Default Type | Valid Types | Description | Sample Output |
|-----------------|--------------|---------------|---|------------------------------|
| Analog | d | d, u, o, x, X | For example, %04d specifies a right-justified decimal value with a minimum field width of 4 digits. The 0 specifies the value is padded with zeros. | 0005 0015 0150 2400 |
| | | | For example, %3u specifies a right-justified unsigned decimal value with a minimum field width of 3 digits. The value is padded with spaces. | 5 15 150 2400 |
| | | | For example, %-3u specifies the same as the example above, except the hyphen (-) before the width specifies the value is left-justified. | 5 15 150 2400 |

- **OVERVIEW**
- *Using Format Specifiers*
-
-

Table 83-3 :Format Specifiers

| Type of Element | Default Type | Valid Types | Description | Sample Output |
|-----------------|--------------|-------------------|---|---|
| Long Analog | ld | d, ld, u, o, x, X | For example, %-7ld specifies a left-justified long decimal value with a minimum field width of 7 digits. The value is padded to the right with spaces. | 5 15 2400 20000 1000000 |
| | | | For example, %05ld specifies a right-justified long decimal value with a minimum field width of 5. A 0 before the width specifies the value is padded with zeros. | 00005 00015 02400 20000 1000000 |
| Floating-point | f | e, f, g | For example, %6.2f specifies a right-justified floating-point value with a minimum total field width of 6 digits. (The decimal point counts as 1 digit.) This means two digits are displayed after the decimal point and at least three digits are displayed before the decimal point. The value is padded with spaces. | 5.51 150.08 24000.65 |
| Message | s | s | For example, %5s specifies a right-justified message string with a minimum field width of 5 characters. The value is padded with spaces. | on off alarm |
| | | | For example, %-5s is the same as the example above, except the hyphen (-) before the width specifies the value is left-justified. | on off alarm |

Alarm Logging Task Definition

Data collected by FactoryLink is stored as a data element in a real-time database. Each time data is collected, the value stored in the real-time database for an element is overwritten by the new data.

Using Distributed Alarm Logging, you can establish criteria that generates an alarm for any defined data element in the real-time database. If the value for the element meets the criteria established for alarming, an alarm message is displayed on the Alarm Viewer screen for the FactoryLink operator. The element is then monitored throughout the alarm cycle in the Alarm Viewer until the element value no longer meets the alarm criteria.

The alarm criteria can be set to require an acknowledgment from you or not. The acknowledgment provides a means to ensure you know the alarm has been generated because the alarm does not clear from the display until it has been acknowledged.

If you want to preserve the time of alarm, alarm message, operator who acknowledged it, and logbook entries for historical purposes you can configure Distributed Alarm Logging to send the alarm data to a disk-based relational database via an Historian. The frequency Distributed Alarm Logging logs data to the relational database depends on the return to normal for each of the elements being monitored.

You can configure Distributed Alarm Logging to distribute the alarm messages along a network if you want the alarms to be viewed on more than one workstation. In this way alarms that relate to multiple operators can be seen across the network. If the alarms are being logged and acknowledged, then your name or the name of the operator who acknowledged the alarm and the node name where it was acknowledged are included in the alarm data sent to the relational database.

This chapter introduces the operational concepts to configure Distributed Alarm Logging operations. Refer to “Distributed Alarm Logging” on page 499 in the *FactoryLink Configuration Guide* for information on procedures to configure this task. Use “Distributed Alarm Logging” on page 107 in the *FactoryLink Reference Guide* for a quick description of this task’s panels, its entries, and its associated error messages.

- **ALARM LOGGING TASK DEFINITION**

- *Alarm Logging Methodology*

-
-

ALARM LOGGING METHODOLOGY

The following steps describe and illustrate how memory-resident real-time data is monitored for alarm conditions and is then logged to a disk-based relational database.

1. The real-time database receives and stores data from various sources, such as a remote device, user input, or computation results from a FactoryLink task.
2. Distributed Alarm Logging reads and compares the values of the data elements stored in the real-time database with criteria defined within Distributed Alarm Logging.
3. When the value of the data element meets the criteria for an alarm, Distributed Alarm Logging sends the alarm to the Application Editor for display on the Alarm Viewer screen.
4. Each time the data value changes, Distributed Alarm Logging evaluates the element and updates the Alarm Viewer through the Application Editor.
5. When the value of the data element no longer meets the criteria for an alarm, Distributed Alarm Logging removes it from the active alarm list and the Alarm Viewer clears.
6. If the alarms are being logged to a relational database, Distributed Alarm Logging sends the alarm data to the relational database via a Historian.

ESTABLISHING THE ALARM CRITERIA

The criteria you establish for an alarm generates an alarm message on the Alarm Viewer screen for the FactoryLink operator. An example of this is the alarming of the pressure in a fuel tank. A safe pressure is established as a reading of 0-1000. If the pressure exceeds 900, a HI alarm flags you of the pending danger. The situation may be critical if the pressure exceeds 950, which would generate a HIHI alarm. In addition, you may wish to configure the system so you can prevent the pressure from ever reaching 900 by setting an alarm for >850.

In all three cases, the condition for the tag is greater than but each alarm is differentiated from the other. As the pressure changes, the display is updated to reflect the new readings and messages. When the pressure drops to 800, the danger passes and the alarms are no longer active.

Three components that the data element value must be checked against establish this alarm:

- **Limit**—The limit is the value the condition is checked on. The example establishes the limit as 900.
- **Condition**—The condition is what sets off the alarm. In the example the condition is exceeds or greater than.
- **Deadband**—The deadband is a range above or below the limit the alarm stays active in. The example uses a deadband of 100 ($900-100 = 800$).

The limit and the deadband can both be set with a constant value or the value from another data element. The conditions a referenced element can generate an alarm under are

| | |
|-----------------|---|
| ON | An alarm is triggered when the value of the element referenced is ON (1). |
| OFF | An alarm is triggered when the value of the element referenced is OFF (0). |
| TGL | An alarm is triggered when the value of the element referenced changes from ON (1) to OFF (0) or from OFF (0) to ON (1). |
| HI GT HIHI or > | An alarm is triggered when the value of an analog or floating-point element is greater than the value specified by the Limit. |
| LO LT LOLO or < | An alarm is triggered when the value of an analog or floating-point element is less than the value specified by the Limit. |

- **ALARM LOGGING TASK DEFINITION**

- *Establishing the Alarm Criteria*

-
-

- GE or >= An alarm is triggered when the value of an analog or floating-point element is greater than or equal to the value specified by the Limit.
- LE or <= An alarm is triggered when the value of an analog or floating-point element is less than or equal to the value specified by the Limit.
- EQ or = An alarm is triggered when the value of an analog or floating-point element is equal to the value specified by the Limit.
- NE or <> An alarm is triggered when the value of an analog or floating-point element is not equal to the value specified by the Limit.

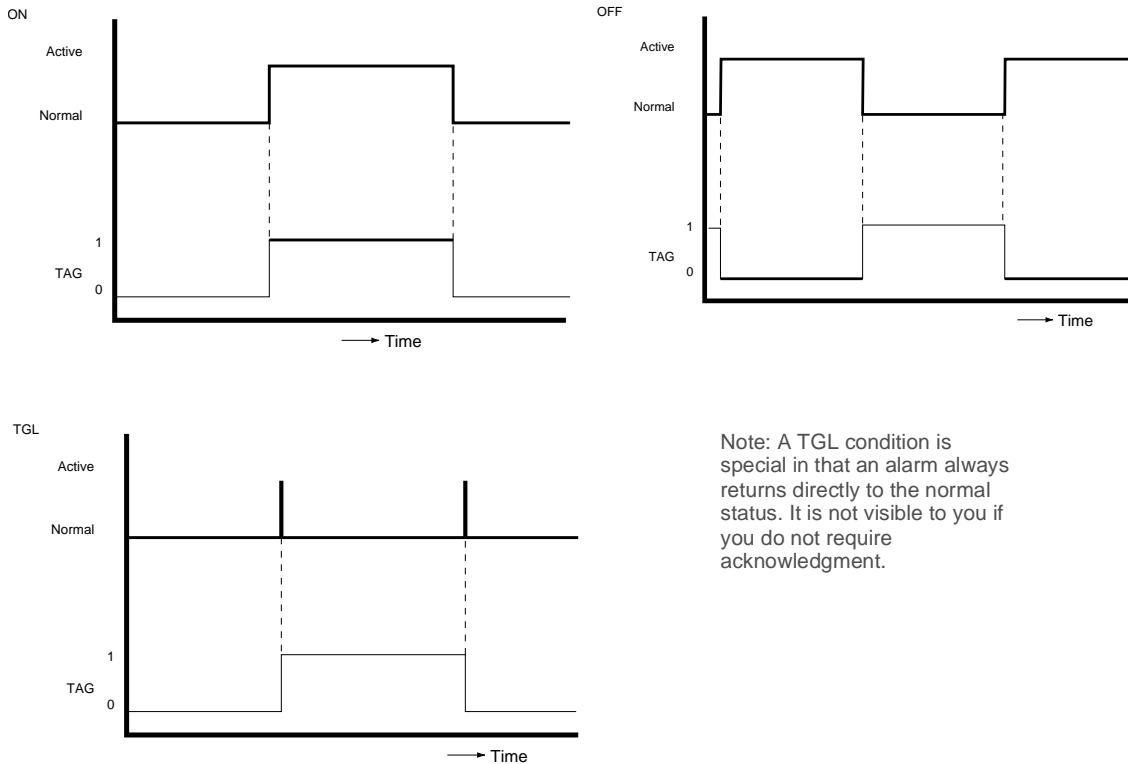
Not all conditions are valid for all tag types. The following table illustrates the conditions supported by each tag type.

| | Digital | Analog | Longana | Float | Message | Mailbox |
|-----------------|---------|--------|---------|-------|---------|---------|
| ON | X | | | | | |
| OFF | X | | | | | |
| TGL | X | X | X | X | X | |
| LOLO LO LT < | | X | X | X | | |
| HIHI HI GT > | | X | X | X | | |
| LE < = | | X | X | X | | |
| GE > = | | X | X | X | | |
| EQ = | | X | X | X | X | |
| NE < > | | X | X | X | X | |

The following diagrams illustrate the behavior of an alarm with specified limits by tag type. The diagrams represent the alarm status active and normal based on the value, limit and deadband.

Digital Elements

Figure 84-1



Note: A TGL condition is special in that an alarm always returns directly to the normal status. It is not visible to you if you do not require acknowledgment.

Analog and Float Elements

The principle of operations are identical when operating on analog, longana or float tag types. The smallest unit detected is dependent on the type.

- **ALARM LOGGING TASK DEFINITION**

- *Establishing the Alarm Criteria*

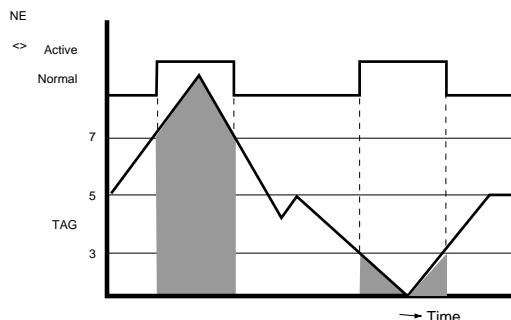
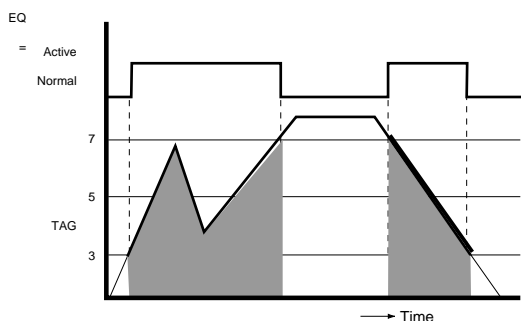
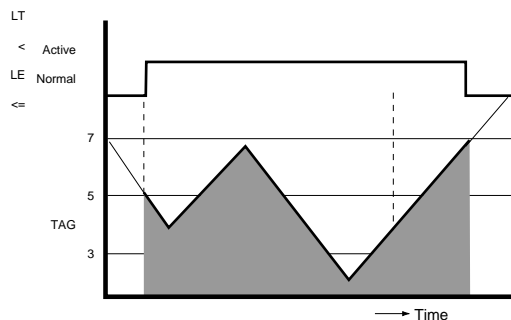
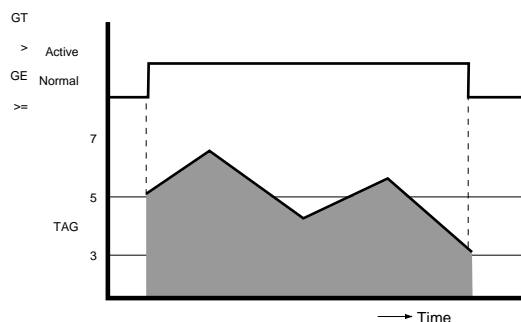
-
-

All examples are based on the following parameters:

Limit: 5

Deadband: 2

Figure 84-2



Message Elements

Message tags can detect a change in the value and be checked on equal or not equal to a limit based on the first 12 characters of the message.

If a TGL condition is established, the alarm vanishes as soon as it is detected because it always immediately returns to normal. If the alarm is marked for an operator to acknowledge it, the alarm is visible until it is acknowledged and then clears from the display. If Distributed Alarm Logging is configured for logging the data to a relational database, it logs as an alarm whether it is set to be acknowledged or not. An example of the use of this option is for logging changed operator names.

ALARM STATES

Every time the value of an alarm element is changed, the new value is evaluated on the alarm criteria.

- If the criteria are not met for an alarm, the value is considered to be in a normal condition.
- If the criteria are met, a new alarm is added to the active alarm list and the value is in an active condition.
- Distributed Alarm Logging maintains a running count of the number of alarms in the active queue in a data element represented by the tag name ALC_ACTCNT.
- If the alarm is already active and the value no longer meets the criteria, it returns to normal.
- If the alarm does not require an acknowledgment, it is removed immediately from the list.
- If the alarm required an acknowledgment and has been acknowledged, it is removed from the list.
- If the alarm requires an acknowledgment and has not been acknowledged, it remains on the list until acknowledged and then is removed.
- The removal of an alarm from the list causes the Distributed Alarm Logger to log the message to the relational database, providing the configuration is set to log alarms.

- **ALARM LOGGING TASK DEFINITION**

- *Alarm Grouping*

-
-

ALARM GROUPING

Alarms can be grouped to facilitate administration or analysis. The following three grouping properties are related to alarms:

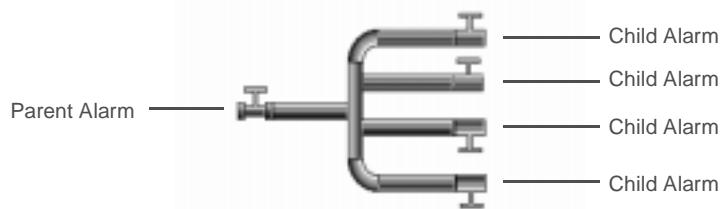
- **Group Name**—The group name is assigned to a class of alarms. Use an uppercase alphanumeric string of 16 characters to define the name. Group names can be identifiers of the severity of the alarm, can group like types such as pressure gauges, or can combine any other characteristics you want.
- **Area**—The area is assigned to each alarm individually. Use an alphanumeric string of 16 characters to specify the area name. More than one alarm can reside in an area and alarms from different groups can also reside together. An area can reflect a physical location such as the boiler room or an area responsibility such as maintenance.
- **Priority**—The priority is a number hierarchy assigned to each individual alarm. Use a number between 1 and 9999 to set priority. Multiple alarms can hold the same priority and multiple groups and areas can have common priority numbers within them.

At least one group must be defined. The use of areas and priorities is optional. Once the grouping properties are assigned, they can be used to filter and sort the alarms on the Alarm Viewer.

PARENT/CHILD RELATIONSHIP

Many times the generation of one alarm causes another alarm to be generated. When these relationships exist, you generally do not want to display the additional alarms resulting from the first alarm. For instance, if the closing of a valve that feeds four different pipelines generates an alarm, it is a reasonable assumption that the lack of flow in each pipe would generate an alarm based on the value of the flowmeter element. These resulting alarms would not be important to you because you already know the flow has been cutoff and why. This relationship between the alarms is identified as a parent/child relationship. In this example the main valve is the parent alarm of each of the flow alarms. The resulting child alarms are not displayed or counted as active alarms because they are a result of the parent alarm.

Figure 84-3



If, on the other hand, the main valve is open and one of the individual pipeline flowmeters registers an alarm, you would want to be advised. In this case the child is not dependent on the parent because the child alarm initiated on its own. This alarm is displayed and counts as an active alarm.

Distributed Alarm Logging allows you to specify parent/child relationships for each alarm so you can determine if a relationship exists that would generate additional alarms you do not want to be advised of. Each alarm can have multiple parent/child relationships. Alarms defined in a remote group can never act as a child alarm. If an alarm has a defined Unique Alarm ID, it can act as a parent on the View node creating the child alarms.

- **ALARM LOGGING TASK DEFINITION**

- *Parent/Child Relationship*

-
-

Each alarm is evaluated by Distributed Alarm Logging and compared to its parent/child relationship prior to displaying.

- If the alarm is a parent, Distributed Alarm Logging displays the alarm.
- If the alarm is a child and the parent is not active, Distributed Alarm Logging displays the alarm.
- If the alarm is a child and the parent is already active, Distributed Alarm Logging determines if the child should be disregarded or displayed based on delay criteria you establish in the relationship.

Within the parent/child relationship two kinds of delays can be specified:

- child alarm delay
- child recovery delay

These delays specify the time allowed between the generation or clearing of a parent alarm and the activation of a child alarm as its own alarm.

Child Alarm Delay

Some relationships are based on the assumption that, when the parent alarm is generated, the child is also generated. Other relationships are based on the assumption that, when the parent alarm is generated, the child is only a result of that alarm, *if* the child is generated within a specified amount of time. If the child does not generate an alarm within this timeframe, the parent/child relationship is disregarded for this alarm invocation. This timeframe is the child alarm delay. If the alarm defined as a child generates an alarm after the allotted timeframe, that alarm is displayed separate from the parent and is counted as an active alarm. Each alarm (parent and child) has to return to normal to clear the alarm states. When both have returned to normal, the parent/child relationship is reestablished and, at the next invocation of the parent, the timer is started again for the child to display or not.

Figure 84-4 Child Alarm Delay - Child Suppressed

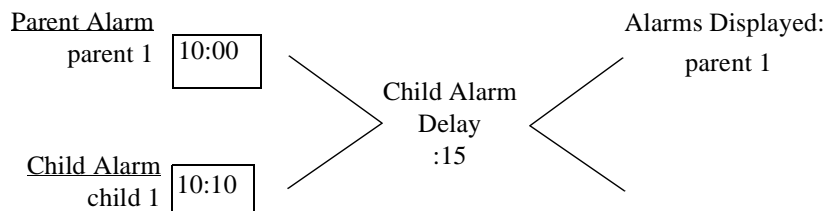
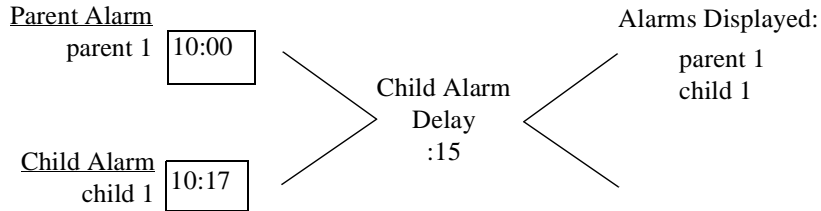
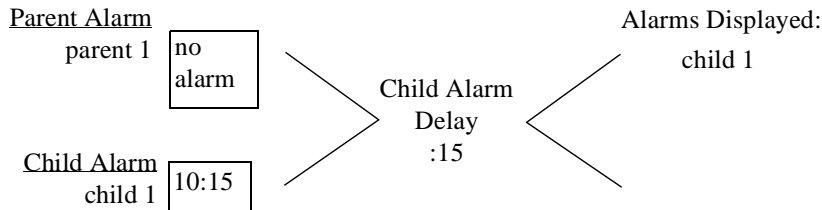


Figure 84-5 Child Alarm Delay - Child Not Suppressed**Figure 84-6** Child Alarm Delay - Child Alarm

Distributed Alarm Logging allows you to determine the maximum length of time between the generation of a parent alarm and the time you feel that relationship is no longer involved in the alarm process. You can define this if you feel the child is always a result of the parent regardless of when it is generated, as long as the parent is in the alarm state.

Child Recovery Delay

Following the same assumption that the parent alarm caused the child alarm, the next assumption is clearing the parent alarm clears the child alarm. In the previous example, the main valve (parent) was shut off. This generated the four pipeline alarms but they are disregarded because they are not important. If the main valve is now turned on, the flow should return to all four pipelines. If one of those lines remains in an alarm state, it is now important to know. The child recovery delay allows the child alarm to activate in this event. After the child has returned to normal, the parent/child relationship is reestablished.

Distributed Alarm Logging allows you to determine the maximum length of time between the return of the parent and child to the normal state. If the parent returns to the normal state and the child does not within the timeframe specified, the child becomes an alarm of its own.

- **ALARM LOGGING TASK DEFINITION**
- *Parent/Child Relationship*
-
-

Figure 84-7 Child Recovery Delay - Child Recovers

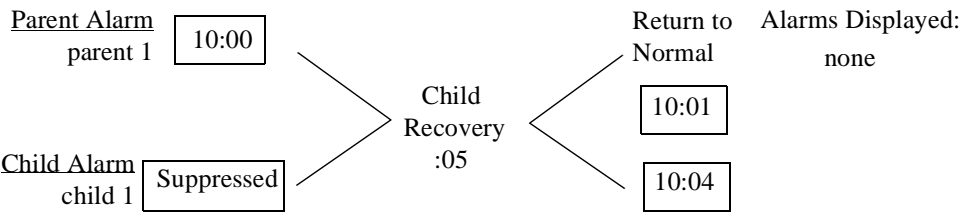
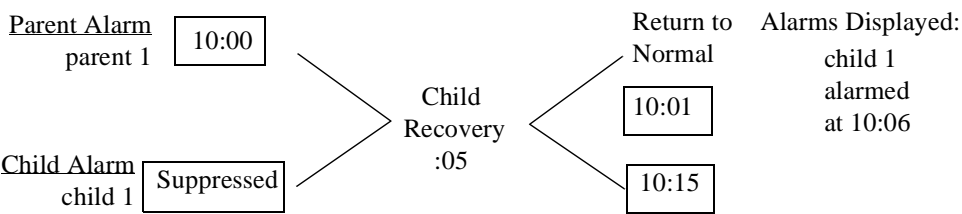


Figure 84-8 Child Recovery Delay - Child Alarms



HIDE ALARMS

You hide alarms most commonly used when you do not need to manage a particular set of alarms. Alarm hiding is used in the following common situations:

- Equipment maintenance
- Redundant systems
- Station functionality
- Bad sensor

Alarm hiding should not be confused with filters used with the Alarm Viewer. Alarm hiding tells the Alarm Logger to disregard all events associated with a particular set of alarms. Alarm filtering only filters the alarms from the viewer, but the alarms are still being logged and tracked.

If an alarm is hidden, it does not act as a parent in parent/child relationships. To avoid potential problems, when the parent alarm is hidden, child alarms must be hidden as well.

Filtering is more common on multi-user or distributed systems. In these architectures, all people have the ability to monitor all alarms; however, certain operators may only be responsible for a subset of these alarms. These operators filter out alarms so they manage only those alarms they are responsible for. The operator login filters at the application level.

Global Hide Tag

The Global Hide tag is used most commonly for redundant systems. In this architecture, one node is the master and all alarms are active for this node (Global Hide tag = 0). The slave node or standby node has the Global Hide tag = 1 until the slave node is to act as the master.

Group Hide Tags

The Group Hide tags are used to hide equipment maintenance alarms. In this case, the developer must be sure to group alarms by machine so, when a maintenance cycle begins, those alarms can be hidden.

The Group Hide tags are also used to define station functionality. This is a special case because a node may have multiple functional requirements. For example, a node may function as a simple operator station for only one piece of equipment one day; and the next, that same node may be the supervisor's station for all of the equipment. Groups are hidden based on the node functionality.

- **ALARM LOGGING TASK DEFINITION**

- *Alarm Persistence*

-
-

Individual Hide Tags

In some systems, individual alarms may need to be hidden to silence an alarm because of a malfunctioning sensor. When the sensor is repaired, the alarm needs to be monitored again.

Remote Groups

No alarm hiding is done on alarms received from remote groups. Alarms should be hidden at the server node. If you do not want to see the alarms, create a filter so the alarms do not show.

Locally Redefined Unique Alarm IDs

For networked systems, if a local Unique Alarm ID definition is created and it matches a remote Unique alarm ID, the masking functions as if the alarm is a local alarm. This applies to global, group, and local alarms.

ALARM PERSISTENCE

Alarm persistence stores the current information about the state of active alarms with the logbook information and the child alarms at user-defined intervals. At startup, the information is read, preserving important information, such as initial time, acknowledgment, and logbook information.

Activating Alarm Persistence

Alarm persistence is activated by placing a `-w` on the program arguments field of Distributed Alarm Logger task line in the SHARED domain. This causes the system to save the alarm logger information to the files:

- {FLAPP}/{FLNAME}/shared/al_log.prs
- {FLAPP}/{FLNAME}/shared/al_log.bak

If the *.prs file is not readable at startup, the *.bak file is read.

The al_log.prs file is updated at the time the Distributed Alarm Logger task is shutdown, on a Persistence Timed Trigger change, or on a Persistence Backup Trigger change. The al_log.bak file is updated on a Persistence Backup Trigger change. Refer to *Core Tasks* of the *FactoryLink Configuration Guide* for more information about the persistence function.

Upon restart of the Distributed Alarm Logger task, the `al_log.prs` or `al_log.bak` file is read into memory and all alarms are checked for validity.

The active alarms are stored using their Unique Alarm ID number. If you have not defined a Unique Alarm ID in the Configuration Manager, one is defined at startup. If the configuration does not change, all alarms receive the same Unique Alarm ID each time at startup. If the configuration changes, however, each alarm ID could be altered and the Distributed Alarm Logger could potentially load persistence information for incorrect alarms or not load persistence information.

ALARM DISTRIBUTION

Using the FactoryLink Local Area Network, task alarms can be distributed over the network. Each node can share one or more groups of alarms with other nodes. The alarm is originated on the node it is defined on and is seen and acknowledged from other nodes that have been configured to receive information on that particular alarm. When the alarm is acknowledged, either at the source or at the remote, the source node accepts the acknowledgment and updates the new alarm status. All nodes receiving information on the alarm are updated.

The distribution is achieved by assigning every node on the network a unique distributed alarm LAN identification number. Distributed Alarm Logging uses this number to identify which nodes receive which alarms. The data exchange then takes place through the use of mailboxes—one send mailbox and one receive mailbox are defined and then the different types of messages between the different nodes are all sent through these two mailboxes.

The different alarm nodes on the network are updated using the broadcast principle of the FactoryLink Local Area Network application. This is achieved by putting all nodes in one FactoryLink LAN group.

Distribution is logically divided into two categories that determine how the alarms are configured at each node:

- **Server** - The server is the node originating the alarms.
- **Client** - The client is the node displaying the alarms. Alarms are acknowledged and logbook messages are created from the client.

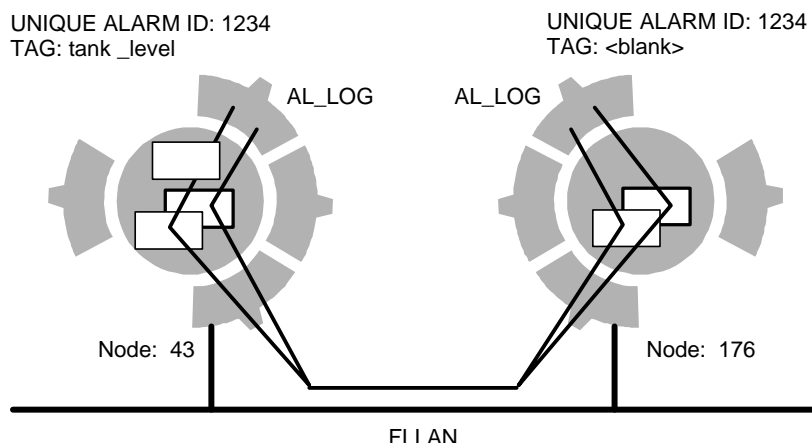
Each node is used as a server and a client or one node can be used as the server of all alarms and the other nodes as clients. Distributed Alarm Logging identifies the node based on the Distributed Alarm Logging LAN ID and matches it with the information on the particular alarm to determine which nodes are performing which duties. The drawing below represents the link between the nodes using the FactoryLink LAN.

- **ALARM LOGGING TASK DEFINITION**

- *Alarm Logging*

-
-

Figure 84-9



The MAXLEN parameters in the LAN Local Names panel must be configured. Set the MAXLEN parameter on all nodes to a value of 64000.

Refer to *Communications of the FactoryLink Configuration Guide* for details on how to enter this information.

ALARM LOGGING

If you want to preserve the time of alarm, alarm data, the operator or node that acknowledged the alarm, and logbook entries for historical purposes, you can configure Distributed Alarm Logging to read data from Distributed Alarm Logging's own elements in the real-time database and send the data to a disk-based relational database or to a text file. Data logged to a relational database is then available for browsing through the FactoryLink Browser.

If a remote group has logging turned on but no database information is defined on the client node, the information is not logged and no error occurs. If a remote node shuts down and restarts or reconnects after a communication failure and the same alarm is still active, the logger tries to insert the alarm into the database twice, generating a Duplicate Entry error.

Distributed Alarm Logging logs data to the relational database in the same way the FactoryLink Database Logger does—in table format via an Historian as the alarm occurs, when the alarm is acknowledged, or after an alarm has returned to the normal state. The tables for alarm logging, one for alarm entries and one for logbook entries, and their associated schemas have already been defined within Distributed Alarm Logging.

The alarm entry table is built using the following schema.

| Column | Type | Array | Description |
|--------|----------|-------|------------------------------------|
| seq | INTEGER | - | Sequence Number of the alarm. |
| aid | INTEGER | - | Alarm ID number. |
| tag | CHAR | 48 | Alarm TAG name. |
| grp | CHAR | 16 | Group alarm belongs to. |
| are | CHAR | 16 | Area alarm belongs to. |
| pri | SMALLINT | - | Priority of the alarm. |
| tim | CHAR | 30 | Initial Time (yearmodyhrmisc.mse). |
| atm | CHAR | 30 | Acknowledge Time (yearmodyhrmisc). |
| ntm | CHAR | 30 | Normal Time (yearmodyhrmisc). |
| dur | INTEGER | - | Alarm Duration in seconds. |
| msg | CHAR | * | Alarm Message. |
| va1 | CHAR | 12 | Variable 1 at Initial State. |
| va2 | CHAR | 12 | Variable 2 at Initial State. |
| va3 | CHAR | 12 | Variable 3 at Initial State. |
| va4 | CHAR | 12 | Variable 4 at Initial State. |
| opr | CHAR | 8 | Operator who acknowledged alarm. |

- **ALARM LOGGING TASK DEFINITION**

- *Logbook*

-
-

* Array is determined by the specified size in the Message Size field of the Alarm Archive Control panel in the Distributed Alarm Logging Setup table.

| Index | Unique | Column list |
|-------|--------|--------------|
| 1 | YES | seq + itime. |

The logbook entry table is built using the following schema.

| Column | Type | Array | Description |
|--------|---------|-------|-------------------------------|
| seq | INTEGER | - | Sequence Number of the alarm. |
| aid | INTEGER | - | Alarm ID number. |
| name | CHAR | 48 | Alarm TAG name. |
| msg | CHAR | * | Logbook entry. |
| opr | CHAR | 8 | Operator who made the entry. |

* Array is determined by the specified size in the Message Size field of the Alarm Archive Control panel in the Distributed Alarm Logging Setup table.

| Index | Unique | Column list |
|-------|--------|-------------|
| 1 | NO | seq. |

LOGBOOK

Distributed Alarm Logging provides a logbook entry capability during run time that allows you to annotate the alarm. These entries are saved in the relational database or a file when the alarms are logged and are also distributed along the network if the alarms are being distributed.

The logbook allows you to add notes to the alarm to document why an alarm occurred, how it was corrected, or what is being done to correct it. If the alarms are being distributed, all operators on the network can add logbook entries to the alarm.

Batch Recipe Task Definition

The FactoryLink Batch Recipe task transfers sets of predefined values, sometimes called recipes, between binary disk files and selected elements and between the real-time database and an external device. In the real-time database, a batch recipe is a collection of elements grouped together for some purpose. These elements can contain internally-generated or operator-entered values.

Perform the following steps to configure a typical Batch Recipe application:

1. Create and animate a graphics display.
2. Complete a Recipe Table.
3. Complete the protocol module Read/Write Table for that external device if you want operator input to change the settings in an external device.
4. Reference the names of these elements in the Read/Write Table for the FactoryLink task communicating with the external device if the task writes element values to registers in an external device.

This chapter introduces the operational concepts to configure Batch Recipe operations. Refer to “Configuring Batch Recipe” on page 577 in the *FactoryLink Configuration Guide* for information on procedures to configure this task. Use “Batch Recipe” on page 17 in the *FactoryLink Reference Guide* for a quick description of this task’s panels, its entries, and its associated error messages.

- **BATCH RECIPE TASK DEFINITION**

- *Principles of Operation*

-
-

PRINCIPLES OF OPERATION

In the real-time database, a FactoryLink batch recipe is a collection of elements grouped together for some purpose. These elements can contain internally-generated or operator-entered values.

You can perform the following functions with Batch Recipe:

- Define up to 8000 different recipe templates, each associated with a virtually unlimited number of files.
- Store batch recipes in disk files so the total number of different recipes stored on a system is limited only by available disk space.
- Store each batch recipe file under a standard file name.
- Specify up to 8000 elements for one batch recipe template.
- Use with any of the five FactoryLink data types: digital, analog, long analog, floating-point, and message.

You can configure Batch Recipe for use in many diverse applications. For example, a program can use a graphic display for the entry of application values and write these values to an external device using the FactoryLink EDI task. Batch Recipe can save these element values in a recipe file so the program can then read the values from the batch recipe file.

Sample applications, each using a single batch recipe template, are

- Producing a particular line of paint. You can use multiple files using the same recipe template to set various hues or colors of the paint being produced.
- Setting up various external devices with different files for days of the week, end of the month, and other schedules.
- Setting up an environment for a testing procedure with various files to establish different sets of testing parameters.

Refer to “Sample Batch Recipe” in the *FactoryLink Configuration Guide* for more information.

You can use batch recipes in conjunction with any FactoryLink task because each FactoryLink task communicates with other tasks through the real-time database.

Batch Recipe executes as a background task. The task does not require operator intervention at run time unless you design the application to require it.

You can configure Batch Recipe to be triggered by events, timers, or operator commands, such as:

- An external device read operation
- A Math & Logic calculation (either Interpreted or Compiled)
- An activity from another station on a network
- Input from the operator (using a keyboard or pointing-device)

Monitor the Run-Time Manager screen to determine the status of Batch Recipe at run time.

FactoryLink saves recipe files when performing a platform-dependent FLSAVE; however, FactoryLink does not save recipe files when performing a platform-independent (multiplatform) FLSAVE.

- **BATCH RECIPE TASK DEFINITION**
- *Creating and Animating a Graphics Display*
-
-

CREATING AND ANIMATING A GRAPHICS DISPLAY

You create and animate a graphics display using the Application Editor adhering to the following specifications:

- Create input fields on the display that allow an operator to select a recipe name and to set or modify values in the recipe.
- Link these input fields to elements.
- Define buttons for read and save triggers for the recipe (such as Open Recipe and Save Recipe).
- Link the buttons to digital elements. (Use the ON action to create the triggers.)
- Define other buttons, such as a button or key to open the new display from the Main Menu and a button or key to return to the Main Menu.
- Link these buttons to digital elements.
- Assign a name to the display.
- Enter a window element name and default drawing name.
- Use the View menu commands to size the window and choose its location on the screen.

Refer to the *Application Editor Guide* for detailed instructions.

Database Browser Task Definition

The FactoryLink Database Browser task works in conjunction with the FactoryLink Historian task to allow an application to access data in a relational database through a browse window. Browser offers the following features:

- Allows relational data in a relational database to be manipulated from within FactoryLink
- Allows an application to send and retrieve data to and from all external database tables, including those created outside of FactoryLink
- Allows you to define elements referenced by Browser in arrays as well as individually

This chapter introduces the operational concepts to configure Database Browser operations. Refer to “Database Browser” on page 473 in the *FactoryLink Configuration Guide* for information on procedures to configure this task. Use “Database Browser” on page 29 in the *FactoryLink Reference Guide* for a quick description of this task’s panels, its entries, and its associated error messages.

- **DATABASE BROWSER TASK DEFINITION**

- *Principles of Operation*

-
-

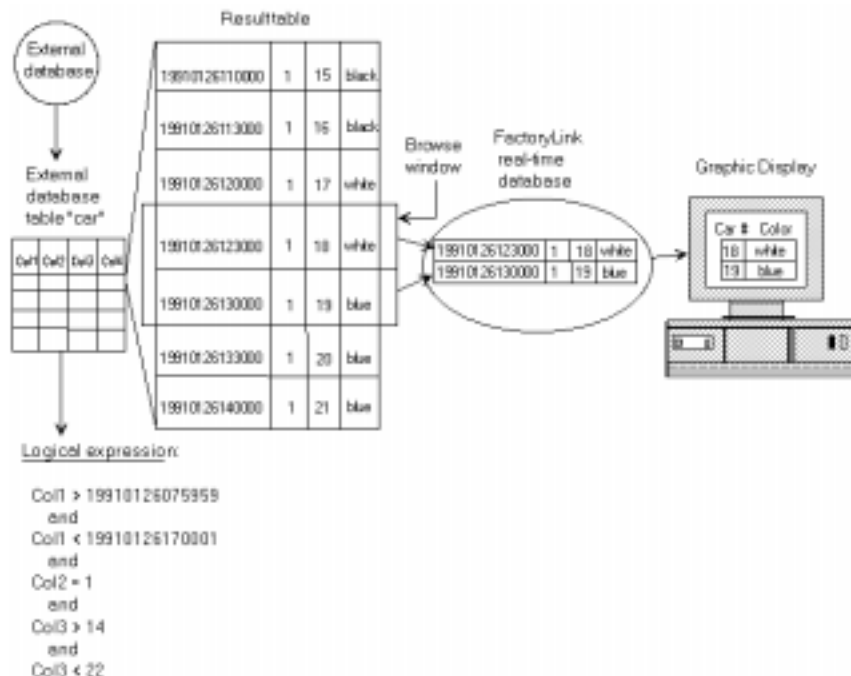
PRINCIPLES OF OPERATION

Browser is a Historian-client task that communicates with Historian through mailbox elements to send and receive historical information stored in an external database.

Browser accesses data in a relational database by selecting the data specified in a configuration table and placing it in a temporary table called a result table. The element views and modifies the data in the result table through a browse window. A browse window is a sliding window that maps data between the relational database and the real-time database. The browse window views selected portions of the result table.

For example, if a graphic screen is used to display the browse window, it can display as many rows of data from the result table as there are elements in the two-dimensional element array. If there are more rows in the result table than in the browse window, the operator can scroll through the result table and see each row of it in the browse window.

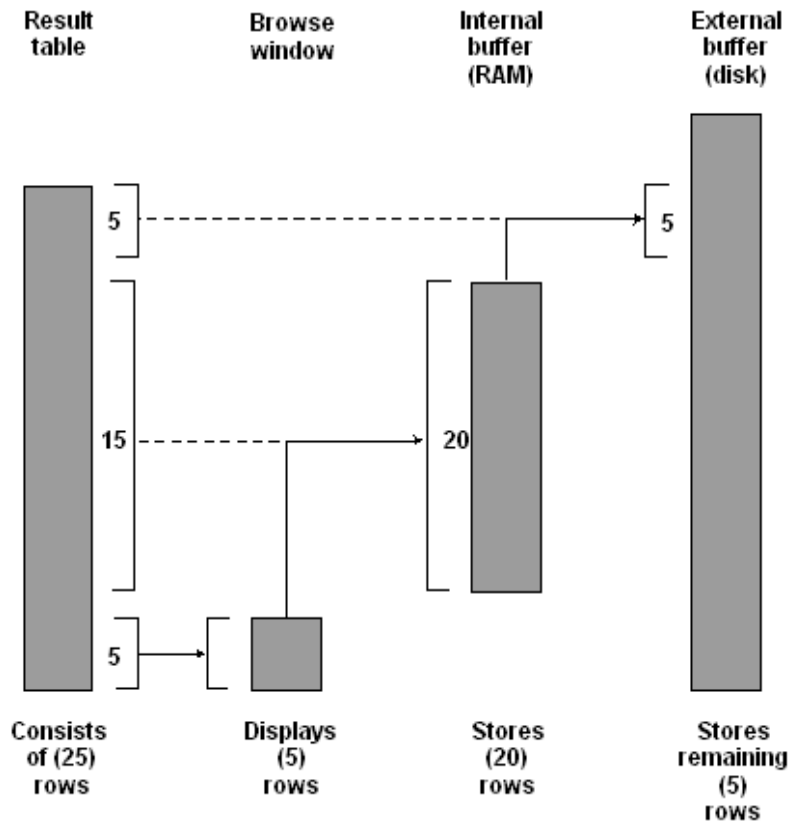
The relationships among the external database, the result table, the browse window, the real-time database, and the graphic display are displayed below.



Browser can read from and write to an entire array of elements in one operation.

An internal buffer stores the rows of the result table in RAM. An external buffer stores the overflow of rows from the internal buffer on disk. This allows the operator to scroll back up through the result table. The buffers are shown in the following illustration.

Figure 86-1



In this example, as the operator scrolls through the result table, the rows of the result table flow into the internal buffer to be stored in memory. Because, in this case, the result table consists of 25 rows and the internal buffer can store only 20 rows, when the internal buffer is full, the excess rows in the internal buffer flow into the external buffer to be stored on disk.

- **DATABASE BROWSER TASK DEFINITION**
- *Use of Logical Expressions*
-
-

USE OF LOGICAL EXPRESSIONS

You use logical expressions to specify the data in a relational database to view or modify. For the purposes of Browser, a logical expression is a command containing a standard Structured Query Language (SQL) WHERE clause. To make a logical expression flexible at run time, use the name of an element whose value is a WHERE clause. If viewing all data from a column in a relational database table, you do not need to specify a logical expression.

You must know how to write a standard SQL statement to configure Browser. See any SQL guide, such as *Quick Reference Guide to SQL* and/or the user manual for the relational database in use for information about writing SQL statements.

To select data from a database table, a logical expression works in conjunction with the table's column name and logical operators to form an SQL WHERE clause. The WHERE clause specifies which rows in a database table to place in the result table. The following table represents part of a sample database table, CAR.

Table 86-1

| TRANDATE | CONVEYOR | CARNUM | COLOR |
|----------------|----------|--------|--------|
| 19910126080000 | 1 | 9 | yellow |
| 19910126083000 | 1 | 10 | red |
| 19910126090000 | 1 | 11 | red |
| 19910126093000 | 1 | 12 | red |
| 19910126100000 | 1 | 13 | silver |
| 19910126103000 | 1 | 14 | black |
| 19910126110000 | 1 | 15 | black |
| 19910126113000 | 1 | 16 | black |
| 19910126120000 | 1 | 17 | white |
| 19910126123000 | 1 | 18 | white |
| 19910126130000 | 1 | 19 | blue |
| 19910126133000 | 1 | 20 | blue |

Table 86-1

| TRANDATE | CONVEYOR | CARNUM | COLOR |
|----------------|----------|--------|----------|
| 19910126140000 | 1 | 21 | blue |
| 19910126143000 | 1 | 22 | brown |
| 19910126150000 | 1 | 23 | burgundy |
| 19910126153000 | 1 | 24 | blue |
| 19910126160000 | 1 | 25 | blue |
| 19910126163000 | 1 | 26 | brown |
| 19910126170000 | 1 | 27 | burgundy |

A sample WHERE clause referencing the previous table CAR is

```
TRANDATE > '19910126075959' AND TRANDATE < '19910126170001' AND
CONVEYOR = 1 AND CARNUM > 14 AND CARNUM <22
```

In this example, the WHERE clause requests:

What colors cars 15 through 21 on conveyor 1 were painted between 8:00 A.M. and 5:00 P.M. on January 26, 1991

From this WHERE clause, the relational database places the following values in a result table.

Table 86-2

| | | | |
|----------------|---|----|-------|
| 19910126110000 | 1 | 15 | black |
| 19910126113000 | 1 | 16 | black |
| 19910126120000 | 1 | 17 | white |
| 19910126123000 | 1 | 18 | white |
| 19910126130000 | 1 | 19 | blue |
| 19910126133000 | 1 | 20 | blue |
| 19910126140000 | 1 | 21 | blue |

- **DATABASE BROWSER TASK DEFINITION**

- *Use of Logical Expressions*

-
-

If the view size of the browse window is 2, the browse window writes the values of the elements in two rows to the real-time database. When the data reaches the real-time database, other FactoryLink tasks can read it and write to it, and an operator can view the data on a graphics screen.

Browser performs four operations:

- **Select**—Uses an SQL select statement to select and retrieve data from a relational database to be displayed in a result table.
- **Update**—Updates the data in the result table and external database. Browser can perform two types of update operations:
 - **Positional**—Updates the current row (row at which the cursor is currently pointing) of data displayed in the browse window.
 - **Logical**—Updates the data described by the logical expression.
- **Insert**—When an update operation cannot be performed because the row to be updated does not exist, inserts a new row of data in the result table.
- **Delete**—Deletes a row from the result table and external database. Browser can perform two types of delete operations:
 - **Positional**—Deletes the current row (row at which the cursor is currently pointing) of data displayed in the browse window.
 - **Logical**—Deletes the data described by the logical expression.

CONFIGURING PROGRAM ARGUMENTS

Configure the following system configuration program arguments to affect Database Browser functionality:

- L or -l Enables logging of errors to the log file. By default, the Database Browser does not log errors.
- N or -n Notifies on the completion of a SELECT trigger that the query resulted in an End of Fetch condition if the rows returned from the query do not equal the rows defined in the View Size. By default, the Database Browser task does not report an End of Fetch condition for a SELECT until a move operation advances the current row past the last row of the query.
- S[4-160] or -s[4-160] Sets the maximum number of SQL statements that the Database Browser has active at one time. The default is 160. For very large applications, this program switch may have to be adjusted if the database server is unable to allocate a resource to open a new SQL cursor.
- V1 or -v1 Writes the SQL statements generated by the Database Browser to the log file. The Database Browser must have logging enabled for this program switch to work. The default is to not write the SQL statements to the log file.
- W[5-300] or -w[5-300] Sets the maximum timeout in seconds for the Database Browser to wait for a response from the Historian task. The default is 30 seconds.

For values less than 30 seconds, this switch will only work correctly when the Historian initially achieved a successful connection with the database server. If the Historian failed to successfully connect with the database server, Database Browser will timeout in 30 seconds regardless of this switch setting.

Perform the following steps to configure one or more arguments:

- 1 Ensure the current domain selected is correct in the Configuration Manager Domain Selection box to modify the Database Browser entry in the System Configuration panel.
- 2 Choose System Configuration on the Configuration Manager Main Menu to display the System Configuration Information panel.
- 3 Enter one or more arguments separated by spaces in the Program Arguments field for the DBBROWSE task.

- **DATABASE BROWSER TASK DEFINITION**
- *Configuring Program Arguments*
-
-

Database Logging Task Definition

FactoryLink stores data it collects and computes as a data element in a real-time database. Each time data is collected or computed, the new data overwrites the element value stored in the real-time database.

Data can be logged to an historical database using Database Logging if data, such as reports or information on trend charts, is to be preserved for historical purposes. Database Logging reads data from the memory-based real-time database and sends it to a disk-based relational database.

Logging sends data to a relational database via FactoryLink Historian. The Historian used for this transfer depends on the relational database receiving the data. This can be either the dBASE IV-compatible database that can be purchased with FactoryLink or a third-party relational database, such as Oracle7 or Sybase. Refer to “Historian Task Definition” on page 115 for details on which third party relational databases are compatible with FactoryLink.

This chapter introduces the operational concepts to configure Database Logging operations. Refer to “Database Logger” on page 347 in the *FactoryLink Configuration Guide* for information on procedures to configure this task. Use “Database Logging” on page 57 in the *FactoryLink Reference Guide* for a quick description of this task’s panels, its entries, and its associated error messages.

- **DATABASE LOGGING TASK DEFINITION**

- *Database Logging Methodology*

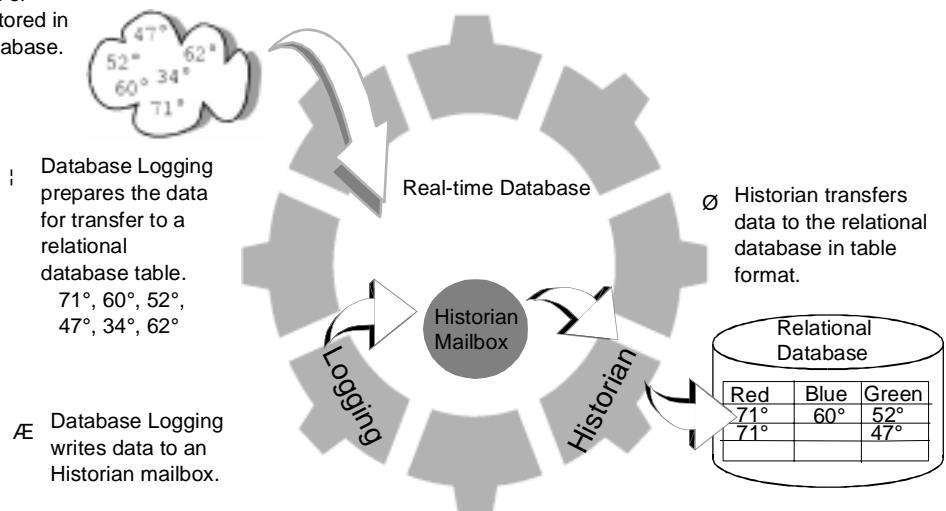
-
-

DATABASE LOGGING METHODOLOGY

The following steps describe and illustrate how memory-resident real-time data is logged to a disk-based relational database:

1. The real-time database receives and stores data from various sources, such as a remote device, user input, or computation results from a FactoryLink task. When data is collected and stored in this database, other tasks can access and manipulate it.
2. Database Logging reads the values of data elements stored in the real-time database and maps the data elements to columns in a disk-based relational database table.
3. Database Logging sends the data from the real-time database to an Historian mailbox in the form of an SQL INSERT statement. The request remains in the Historian mailbox until Historian processes the request.
4. Once Historian processes the request, it connects to the relational database and inserts the data in the relational database file. Once in this file, other applications can use the data.

• Data is collected or computed and stored in the real-time database.



When selecting a logging method, you must consider:

- **Grouping method**—Although you can theoretically log grouped data without a sequence number, the examples show use of a sequence number because this is more typical for grouped data.
- **Sequencing or ordering method** (both available with all grouping methods)
 - Integer
 - Time-stamping
- Indexing
- Deleting method

Three logging methods are available to choose from based on these considerations:

- **Nongrouped/nonsequenced**—Rows of data not associated with a group and with no ordering.
- **Nongrouped/sequenced**—Rows of data not associated with a group and ordered by time-stamping or integer or both.
- **Group name and/or subgroup number grouped** —Rows of data associated with a group name, group name and subgroup number, or subgroup number.

The following sections provide a brief description of each method.

- **DATABASE LOGGING TASK DEFINITION**
- *Grouping Data*
-
-

GROUPING DATA

FactoryLink permits you to log data with or without a group association. If you elect to log data without a group association (nongrouped), the table delineates the data in the table from data in other tables. For example, to maintain the start date for employees at two company locations, such as Dallas and Atlanta, you log the employees located in Dallas in a different table than the employees located in Atlanta.

| | | | | |
|-----------------|---|-----------------|------------------|-------------------|
| Dallas table | { | Lastname | Firstname | Start_date |
| | | Smith | Charlie | 12/14/93 |
| | | Johnson | Jimmy | 2/14/95 |

| | | | | |
|------------------|---|-----------------|------------------|-------------------|
| Atlanta table | { | Lastname | Firstname | Start_date |
| | | Jones | Susan | 11/6/95 |

Grouping data allows you store data with a common structure in the same table but still separate it based on some criteria, such as company location. If you elect to associate logged data with a group, the table structure must include a column that contains the group associated with the row of data.

| | | | | | |
|-----------------|---|-----------------|-----------------|------------------|-------------------|
| Group column | → | Location | Lastname | Firstname | Start_date |
| | | Dallas | Smith | Charlie | 12/14/93 |
| | | Atlanta | Jones | Susan | 11/6/95 |
| | | Dallas | Johnson | Jimmy | 2/14/95 |

One of the following specifies the group associated with a data row:

- **groupname**
 where
groupname Is a unique alphanumeric name you assign that does not typically change. For example, if you want to group data by company location, which does not typically change, you group data by *groupname*.
- **groupname_subgroupnum**
 where
groupname Is a unique alphanumeric name you assign and *subgroupnum* is a number Database Logging assigns.
subgroupnum is a suffix of *groupname* preceded by an underscore as in Shift_1. Database Logging creates new subgroups during run time based on a defined cycle.

Use this method if you want to subgroup data by group. For example, if you want to group data by company location and employment year, use *groupname_subgroupnum* to group data.

Group column →

| Location | Lastname | Firstname | Start_date |
|------------|----------|-----------|------------|
| Dallas_93 | Smith | Charlie | 12/14/93 |
| Atlanta_95 | Jones | Susan | 11/6/95 |
| Dallas_95 | Johnson | Jimmy | 2/14/95 |

- **DATABASE LOGGING TASK DEFINITION**
- *Grouping Data*
-
-

- *subgroupnum*
where

subgroupnum Is a number Database Logging assigns. Logging creates new subgroups during run time based on a cycle you define.

Use this method if you want group assignments to periodically increment by number. For example, if you want to group data by employment year, you can group data by subgroup number.

Group column
→

| Location | Lastname | Firstname | Start_date |
|----------|----------|-----------|------------|
| 93 | Smith | Charlie | 12/14/93 |
| 95 | Jones | Susan | 11/6/95 |
| 95 | Johnson | Jimmy | 2/14/95 |

Subgroups without an associated group name are by default associated with a null or blank group. All subgroups created without a group name belong to this null group. This is important to remember when deleting null group data. Refer to “Deleting Group Data” on page 85 for details.

ORDERING DATA

FactoryLink permits you to log grouped and nongrouped data with or without a sequence or order number. If you elect to associate a sequence number with rows of data, your table structure must include a column that contains the sequence number associated with the row of data (record). The sequence number can be an integer or a time-stamping.

Integer

Choose an integer for event data logging. The following table illustrates the use of an integer to order the data.

Reflects order collected →

| Seq_Int | Lastname | Firstname | Start_date |
|---------|----------|-----------|------------|
| 1 | Smith | Charlie | 12/14/93 |
| 2 | Jones | Susan | 11/6/95 |

If grouped data is logged with an integer sequence number, each group has a unique count; that is, the first record for each group is assigned number 1, the second record number 2, and so on.

Any data logged with an integer sequence number without a group association belongs to the null or blank group. This is important to remember when deleting null group data. Refer to “Deleting Group Data” on page 85 for details.

Time-stamping

Choose time-stamping if you want to associate data with a specific time. The following table illustrates the use of a time-stamping to order the data.

Reflects order collected →

| Seq_Time | Lastname | Firstname | Start_date |
|-----------|----------|-----------|------------|
| 486552600 | Smith | Charlie | 12/14/93 |
| 486552605 | Jones | Susan | 11/6/95 |

FactoryLink time-stamping is tied to time maintained by an internal clock based on the global tag, SECTIME. This tracks time from the starting point at midnight January 1, 1980 in intervals of one second.

- **DATABASE LOGGING TASK DEFINITION**
- *Indexing*
-
-

INDEXING

Records are displayed in a relational database table in the order they are added. This is known as the natural or physical order. The field or fields selected for indexing are called the index key. When a table is indexed, an index file is created that contains the indexed order. The actual record order in the database file remains unchanged. Accessing indexed records is faster than accessing non-indexed records.

In some cases, you may want to retrieve the records in a different order, such as alphabetical order. This is achieved by creating an index for the table.

An index reorganizes the records in a database file by the values in one or more of the fields. Index the records by the Lastname field to reorganize the records in the example table in alphabetical order.

Index by last name
(index key)

| Location | Lastname | Firstname | Start_date |
|----------|----------|-----------|------------|
| Dallas | Johnson | Jimmy | 2/14/95 |
| Atlanta | Jones | Susan | 11/6/95 |
| Dallas | Smith | Charlie | 12/14/93 |

You can index on more than one column. Records are sorted on the first column specified, then the next, then the next. Index the records by the Location field plus the Lastname field to reorganize the records in the example table in alphabetical order within a company location.

Index by last name
and location
(index key)

| Location | Lastname | Firstname | Start_date |
|----------|----------|-----------|------------|
| Atlanta | Jones | Susan | 11/6/95 |
| Dallas | Johnson | Jimmy | 2/14/95 |
| Dallas | Smith | Charlie | 12/14/93 |

INDEXING FOR THE TREND TASK

You must specify a sequence number as the index or part of the index if you want to include logged data on a Trend chart. What you specify as the index depends on the type of Trend chart you are using to display the data:

- Time-based chart—Must index on a time-stamping sequence number.
- Event chart—Must index on an integer sequence number.

Time-Base Chart Indexing

You must specify a time-stamping sequence number as the index key to display logged data on a time-based chart.

Use this column as index key →

| Seq_Time | Lastname | Firstname | Start_date |
|-----------|----------|-----------|------------|
| 486552600 | Smith | Charlie | 12/14/93 |
| 486552605 | Jones | Susan | 11/6/95 |
| 486552710 | Johnson | Jimmy | 2/14/95 |

Event Chart Indexing

You must specify an integer sequence number as the index key to display logged data on an event chart.

Use this column as index key →

| Seq_Int | Lastname | Firstname | Start_date |
|---------|----------|-----------|------------|
| 1 | Smith | Charlie | 12/14/93 |
| 2 | Jones | Susan | 11/6/95 |
| 3 | Johnson | Jimmy | 2/14/95 |

- **DATABASE LOGGING TASK DEFINITION**
- *Indexing for the Trend Task*
-
-

Event and Time-based Trend Chart Indexing

You must include both an integer and a time-stamping sequence number in two separate columns and specify two indices: one with the integer and the other with the time-stamping to display the same logged data on both an event and a time-based trend chart. Refer to “Indexing” on page 80 for details on indexing.

Use each column as one index key

| Seq_Int | Seq_Time | Last | First | Start_date |
|---------|-----------|---------|---------|------------|
| 1 | 48652600 | Smith | Charlie | 12/14/93 |
| 2 | 486552605 | Jones | Susan | 11/6/95 |
| 3 | 486552710 | Johnson | Jimmy | 2/14/95 |

Grouped Data Indexing

You must specify an index key that includes both the sequence number and the group column to display grouped data on an event chart. The sequence number column must contain an integer. The group column contains the group association of *groupname*, *groupname_subgroupnum*, or *subgroupnum*.

Use these two columns as index key

| Seq_Int | Location | Lastname | Start_date |
|---------|----------|----------|------------|
| 1 | Atlanta | Jones | 11/6/95 |
| 2 | Dallas | Johnson | 2/14/95 |
| 3 | Dallas | Smith | 12/14/93 |

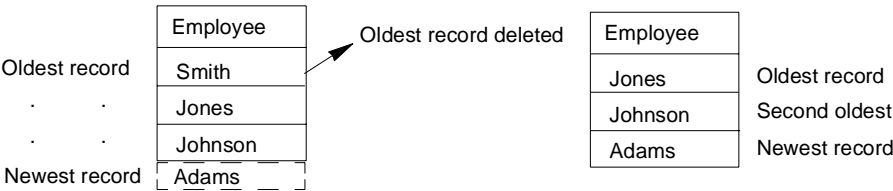
DELETION METHODS

The logging method you choose depends in part on how data gets deleted from the relational database. Your options include:

- Record rollover
- Subgroup rollover
- Group data deletion

Record Rollover

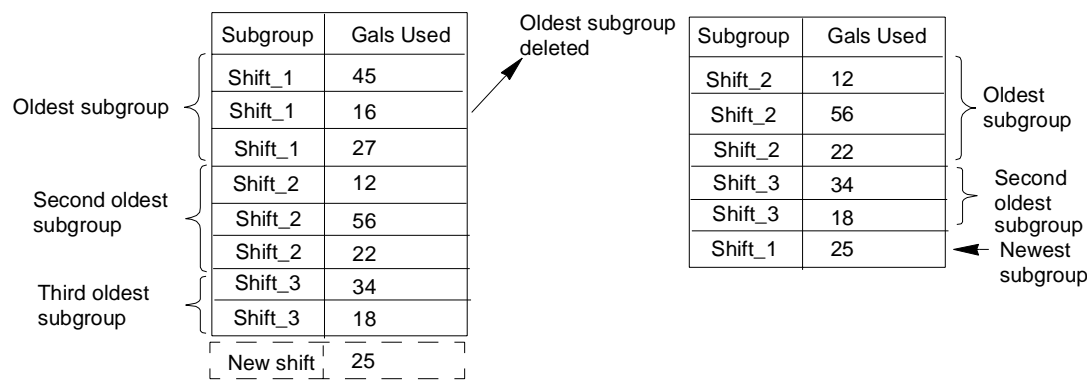
Record rollover is available if you are logging nongrouped data to a FactoryLink dBASE IV relational database. Record rollover permits you to specify the maximum number of records allowed in the database file. Once this maximum is reached, the next attempt at logging a record overwrites the oldest record in the database. If the database file permits only three records, it overwrites the oldest record when the fourth record is logged.



- **DATABASE LOGGING TASK DEFINITION**
- *Deletion Methods*
-
-

Subgroup Rollover

Subgroup rollover is available if you are logging by group/subgroup or subgroup. Subgroup rollover permits you to specify the maximum number of subgroups allowed in the database file. Once the maximum is reached, the next attempt to create a new subgroup overwrites the oldest subgroup in the database. The subgroup number assigned to the oldest subgroup becomes the subgroup number for the new subgroup.



For example, the group name is Shift and allows three subgroups before subgroup rollover occurs. A new subgroup is created every eight hours.

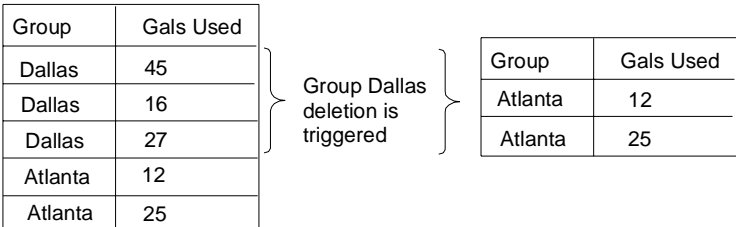
- At 6:00 A.M., Shift_1 is created and data is logged to Shift_1 until 2:00 P.M.
- At 2:00 P.M., Shift_2 is created and data is logged to Shift_2 until 10:00 P.M.
- At 10:00 P.M., Shift_3 is created and data is logged to Shift_3 until 6:00 A.M.

At 6:00 A.M. subgroup rollover occurs, creating a new Shift_1 and deleting the data for Shift_1 from the previous day. This eliminates the need to manually delete data no longer required.

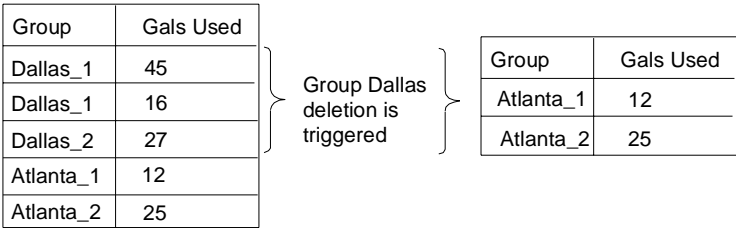
Deleting Group Data

If you are logging grouped data or nongrouped sequenced data, delete all the records for a group in a single operation using a group delete trigger.

The following figure illustrates when group data for Dallas is deleted, the group data for Atlanta remains:



The following figure illustrates when a group delete is triggered, creation of new subgroups begins counting at one:



If you specify the null group or leave the group name blank for the delete group operation, the following data is deleted from the relational database table:

- All subgroups without a *groupname* association
- All nongrouped records sequenced by an integer

Use multiple tables to store the data if you do not want this to happen.

- **DATABASE LOGGING TASK DEFINITION**

- *Database Logging Methods*

-
-

DATABASE LOGGING METHODS

After considering the available database logging methods, choose the one best suited to your application.

Nongrouped/Nonsequenced Method

Use this method if you want to log data without a group association in no particular order. Consider the following benefits and restrictions before selecting this method:

- The number of records in a table are controlled using record rollover if you are logging to a FactoryLink dBASE IV table.
- Data logging is faster than other methods.
- Data is only used by a FactoryLink task or third-party software that does not require an index based on a sequence number.
- Data retrieval is slow compared to other methods.

Nongrouped/Sequenced Method

Use this method if you want to log data without a group association and in a specific order. Time-stamping or integer reflect the order. This time-stamping or integer can be used as a unique index. Consider the following benefits and restrictions before selecting this method:

- The number of records in a table can be controlled using record rollover, if you are logging to a dBASE IV table.
- Data can be indexed by a sequence number, either time-stamping or integer.
- All nongrouped/sequenced by integer records can be deleted with a single operation.
- No ability to delete a data subset.

Grouped method

Use this method if you want to log data with a group and/or subgroup association. Grouping is best for situations when the group name does not change frequently. If you need subgroups with the group name, Database Logging creates the subgroups as they are needed. Use the subgroup method when logging data with a group association and the group name changes frequently and you do not want to assign a unique name to each group.

Consider the following benefits and restrictions before selecting a grouping method:

- Each group must be assigned a unique name.
- Database Logging creates subgroups.
- Data is indexed by group column and sequence number, either time-stamping or integer.
- All records for a group/subgroup can be deleted with a single operation.
- You cannot restrict the number of records in a group/subgroup.
- All records for the oldest subgroup can be deleted with a single operation using subgroup rollover.
- You can control the number of subgroups allowed.

- **DATABASE LOGGING TASK DEFINITION**
- *Configuring Program Arguments*
-
-

CONFIGURING PROGRAM ARGUMENTS

You can configure the following system configuration program arguments to affect Database Logging functions:

- E or -e This program switch causes Database Logging to set the completion trigger when the Historian task processes the logging operation. By default the completion is set when Database Logging sends the request to the Historian mailbox. With this switch, the completion trigger for all log operations means the Historian task has processed the logging transaction.

Setting the completion trigger does not guarantee the log transaction is successful; it only means the log transaction has completed.
- L or -l This program switch enables error logging to the log file. By default Database Logging does not log errors.
- Q or -q [100-2000] This program switch sets the maximum number of outstanding asynchronous logging transactions the Historian task has not completed. Once this limit is reached, Database Logging operates synchronously until the number of uncompleted logging transactions is reduced. By default Database Logging allows for 100 outstanding logging transactions before operating in a synchronous mode.
- S### or -s### This program switch sets the maximum number of SQL statements Database Logging has active at one time. The default is 30. Valid values are one to whatever the database server the Historian is connected to allows.
- W or -w[5-300] This program switch sets the maximum timeout in seconds for Database Logging to wait for a response from the Historian task. The default is 30 seconds.

For values less than 30 seconds, this switch will only work correctly when the Historian initially achieved a successful connection with the database server. If the Historian failed to successfully connect with the database server, Database Logging will timeout in 30 seconds regardless of this switch setting.

Perform the following steps to configure one or more arguments:

- 1 Choose System Configuration on the Configuration Manager Main Menu to display the System Configuration Information panel.

- 2 Enter one or more arguments separated by spaces in the Program Arguments field for the DBLOG task.

- **DATABASE LOGGING TASK DEFINITION**
- *Configuring Program Arguments*
-
-

Data-Point Logging Task Definition

FactoryLink stores the data it collects or computes as a data element in a real-time database. Each time a new value for an existing real-time database element is collected or computed, the current value stored in the real-time database for that element is overwritten by the new data.

You log data to an historical database using a Logger task if you want to preserve data for historical purposes. FactoryLink has two loggers: Data-Point and Database. Both tasks read data from the real-time database and send it to the disk-based relational database via a FactoryLink Historian.

The Historian used for this transfer depends on the relational database receiving the data. The database can be either the dBASE IV-compatible database that can be purchased with FactoryLink or a third-party relational database, such as Oracle. Refer to “Historian Task Definition” on page 115 for details on which third party relational databases are compatible with FactoryLink.

Data-Point Logging simplifies the task of logging data by providing pre-configured tables. Because the tables are pre-configured, Data-Point Logging can only be used to store shared, numeric value tags. For all other types of tags, Database Logging should be used. Only one tag is logged to the database at a time. Multiple tags can be recorded to the same database and sorted later, if necessary. If information about the state of multiple tags at a particular trigger point is desired, the Database Logging task should be used. If you are uncertain whether to use Data-Point or Database Logging for your task, refer to “Overview” on page 21 for an explanation and comparison of both logging tasks.

This chapter introduces the operational concepts to configure Data-Point Logging operations. Refer to “Configuring Data-Point Logging” on page 383 in the *FactoryLink Configuration Guide* for information on procedures to configure this task. Use “Data-Point Logging” on page 83 in the *FactoryLink Reference Guide* for a quick description of this task’s panels, its entries, and its associated error messages.

- **DATA-POINT LOGGING TASK DEFINITION**
- *Data-Point Logging Function*
-
-

DATA-POINT LOGGING FUNCTION

Data-Point Logging reduces the task of configuring data to be logged. Data-Point Logging uses a structured database with the number of columns and the names of the columns predefined by the Data-Point Logging Schema task. Because the table structures are pre-configured, the Data-Point Logging task can only be used to log shared, numeric value tags. The tags to be logged can be specified in the Configuration Manager by means of the Data-Point Logging Information panel or the Application Editor Tag Definition dialog. Refer to “Data-Point Logging Table Schema Configuration” in the *FactoryLink Configuration Guide* and the *Application Editor Guide* for details.

Preconfigured Data-Point Logging Tables

Data-Point Logging provides five default data-point logging tables, each mapped to a specific default schema name, to provide for efficient storage and retrieval of certain tag types. These log tables are

Table 88-1 Pre-configured Data-Point Logging Tables

| Table | Used to Store: |
|----------|---|
| FLOATVAL | floating value data |
| LONGANA | long analog (large integer) data |
| ANALOG | analog (small integer) data |
| LOGDATA | general log data |
| TRENDATA | data that will be used for trend analysis |

Each preconfigured data-point logging table uses the Database Alias Name **DATALOG** which references the relational database where Historian sends the data from Data-Point Logging. In addition, each default table refers to the Historian Mailbox mailbox tag entry **DB4HISTMBX**.

Data-Point Logging allows you to define your own data-point logging tables if you need one other than the pre-defined tables. For details, see “Data-Point Logging Table Schema Configuration” on page 384.

Preconfigured Data-Point Logging Table Schemas

Data-Point Logging provides four default data-point logging table schemas, each accepting a different logged data type.

Table 88-2 Default Data-Point Logging Table Schemas

| Schema Name | Data Type |
|-------------|--------------------------------------|
| TAGDATA | float |
| SMALLINT | smallint (analog, data) |
| LARGEINT | integer (longana, analog, data) |
| FLOATVAL | float (float, longana, analog, data) |

The maximum records allowed in a dBASE IV database table governed by any of the four default Data-Point Logging table schemas is 1,000,000. Each default schema also specifies a maximum tagname column width of 48.

You must specify a schema for the table in the Data-Point Schema Control panel if you define your own Data-Point Logging table. Refer to *Logging* in the *FactoryLink Configuration Guide* for more information.

Data Logged

Data is logged to a predefined table structure. For each event logged, the database row reflects the following entries.

| LOGTIME | TAGNAME | TAGVALUE |
|---------|---------|----------|
| | | |

- LOGTIME is a numeric column that stores the time the tag was logged
- TAGNAME is the name of the tag that was logged at LOGTIME.
- TAGVALUE is the value of the tag logged at LOGTIME

- **DATA-POINT LOGGING TASK DEFINITION**

- *Logging Methods*

-
-

Only one tag and the value of the tag is recorded in each row. This means less data is stored and captured at each logging trigger, optimizing database storage space. The data-point logging configuration also allows you to change the length of the tagname column and the column type and precision of the tag value column for further maximizing of storage space.

The index created for Data-Point Logging tables is a multi-column index based on Tag Name and Log Time with Tag Name the major key.

Data-Point Logging also allows you to dynamically add or remove tags from the list of tags being logged during run time without affecting the database structure.

LOGGING METHODS

With Data-Point Logging, you can specify when a tag (data point) is to be logged based on one or more of the following:

- A change in the tag (exception logging)
- A fixed-time interval
- A change in a trigger tag

If a tag is logged more than once during a given second, any values logged after the first occurrence within that second are ignored.

At task startup, all exception and fixed-time interval tags are logged to create a default beginning reference point. Triggered logging tags are not logged at startup because the trigger tag is not initiated yet.

Logging Based on a Change in the Tag Value (Exception Logging)

You can configure Data-Point Logging to log an event whenever the value of the selected tag changes. FactoryLink polls for values of all tags specified for Data-Point logging. If the value has changed from the previous time it was logged, another event is recorded to the log.

Filtering Exception-Logged Tags

If a given tag changes frequently but not all changes are significant, you can configure deadbanding on the tag so only significant changes are logged. This reduces the data logged and decreases system processing time. This feature is called deadbanding, which allows you to specify a dead band around a tag value for each tag not considered significant enough to record the changed value to the system. This band can be an integer or a percent of value. Refer to the *Core Task*

Section of the [FactoryLink Configuration Guide](#) for a detailed description of the deadbanding function and how to configure it.

Logging Based on a Fixed-Time Interval

You can configure Data-Point Logging to log an event at a specific time interval with the minimum interval being one second. Fixed-time-interval logging rates are based on either seconds, minutes, hours or days, as calculated from a starting point of midnight. It may be appropriate to log a tag value every hour or once a day, depending on the tag.

Fixed-time interval-based logging is tied to time maintained by an internal clock based on the SECTIME global tag. This tag tracks time from the starting point of midnight, January 1, 1980, in intervals of one second.

Logging Based on a Change in a Trigger Tag

You can configure Data-Point Logging to log the value of a tag whenever a specified trigger tag is triggered. Whenever the trigger tag is triggered, Data-Point Logger logs the event.

LOGGING DATA

Data-Point Logging allows you to specify tags to be logged when you are configuring the system and dynamically during run time.

At initialization Data-Point Logging determines which file is newer:

- The configuration table (CT) file, {FLAPP}\shared\ct\dplogger.ct
- The Data-Point Save file, {FLAPP}\log\dplogger.dyn
- The Data-Point Save file specified in the Dynamic Logging Control panel Command File Tag field.

The newer file becomes the list of all tags currently configured for logging. If the Command File Tag is not configured or if the tag contains an empty string, the default Data-Point Save file is used.

- **DATA-POINT LOGGING TASK DEFINITION**

- *Logging Data*

-
-

Logging Data During Configuration

When FactoryLink is being configured, the Data-Point Logging function can be configured for a specific tag in one of two methods:

- Through the Tag Definition dialog displayed in the Application Editor when a new tag is referenced. Refer to the *Application Editor Guide* for more details.
- Through the Data-Point Logging Information panel accessible through the Configuration Manager Main Menu. Refer to “Configuring Data-Point Logging” in the *FactoryLink Configuration Guide* for more information.

Logging Data Dynamically

Data-Point Logging allows you to dynamically add tags to and remove tags from its list of tags to be logged during run-time operation in one or both of the following ways:

- Generate and load a Data-Point Save File that contains one or more logging requests
- Enter a single logging request in a tag

Data-Point Save File

The Data-Point Save File {FLAPP}\log\dplogger.dyn contains a list of all tags currently configured for logging. You can create a Data-Point Save File that is loaded whenever its associated Read trigger is set. The load process causes the list of tags currently being logged to be overwritten by the list of tags specified in the designated Data-Point Save File.

Single Logging Request

Data-Point Logging allows you to enter a single logging request by means of the Command Tag defined in the Dynamic Logging Control panel. This type of dynamic logging request either adds tags to or removes tags from the list of tags currently configured for logging. Optionally, the logging request can have a tag associated with it that describes the logging request status.

This type of dynamic addition and removal of tags is temporary so, every time Data-Point Logging is restarted, the new tag list generated from the Data-Point Save File or the Configuration Table file supersedes the existing list.

Dynamic Data Exchange (DDE) Task Definition

Dynamic Data Exchange (DDE) is a protocol used in Microsoft Windows to transfer data from one application to another.

This chapter contains information about the following topics:

- Principles of Operation
 - DDE Server
 - DDE Client
- DDE Conversations
 - DDE Messages
- Setting Up DDE Server
- Establishing a DDE Link
 - Entering a DDE Formula
 - Modifying a Linked Element
 - Using a Create Link Option
- DDE Client Functions
 - Read Operations
 - Write Operations

This chapter introduces the operational concepts to configure Dynamic Data Exchange (DDE) operations. Refer to “Dynamic Data Exchange (DDE)” on page 667 in the *FactoryLink Configuration Guide* for information on procedures to configure this task. Use “DDE Client” on page 97 in the *FactoryLink Reference Guide* for a quick description of this task’s panels, its entries, and its associated error messages.

- **DYNAMIC DATA EXCHANGE (DDE) TASK DEFINITION**

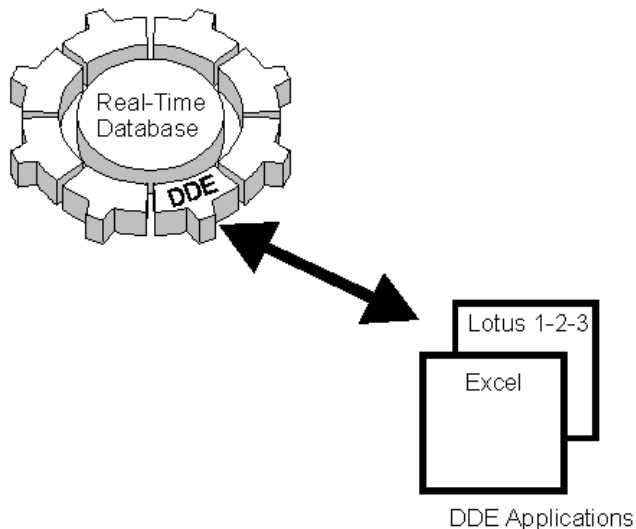
- *Principles of Operation*

-
-

PRINCIPLES OF OPERATION

Dynamic Data Exchange (DDE) is a form of interprocess communication used by many Windows applications to automatically update data and to regulate how and when data is passed between applications. Because DDE protocol is well-defined, two applications that follow the protocol can communicate with each other even though neither application was explicitly written to work with the other.

DDE allows you to perform data display techniques in any software package supporting DDE using FactoryLink data, as shown in the following illustration.



The FactoryLink IV system provides both DDE Server and DDE Client functionality.

DDE Server

Use the FactoryLink DDE Server to establish communications between the FactoryLink real-time database and client software products, such as Lotus 123 and Microsoft Excel.

You may set up DDE links for the FactoryLink DDE Server to be initiated from any external Windows application that offers DDE support; therefore, DDE applications, such as Lotus or Excel, can pass data to or retrieve data from FactoryLink. This means the FactoryLink real-time database can display and modify spreadsheet data, word processing files, and other Windows application data.

The DDE Server also allows client applications to request, modify, advise, and receive notification of updates to elements in the FactoryLink real-time database.

DDE Client

Although the DDE Server is the preferred method for communicating with external DDE applications, you can also use DDE Client to configure bidirectional communication between the real-time database and one or more Windows DDE servers.

You can configure a single FactoryLink system for multiple external DDE servers simultaneously. You setup the FactoryLink DDE Client in the Configuration Manager Main Menu and initiate communication from a FactoryLink application.

DDE CONVERSATIONS

DDE data exchanges occur via DDE conversations. These DDE conversations consist of a DDE server and a DDE client.

DDE servers supply information for client applications; thus, the FactoryLink DDE Server acts as an information source for client applications, such as Excel and Lotus.

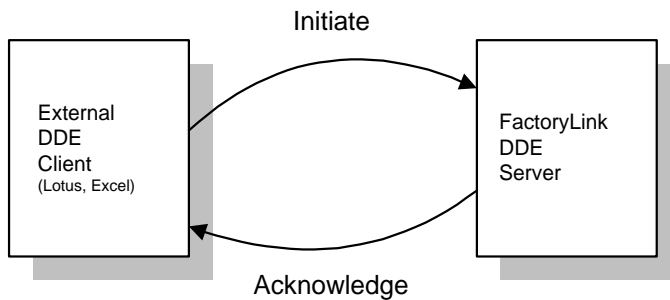
Client applications control the DDE conversation and initiate all data requests. You can set up DDE links in external client applications to retrieve data from the real-time database via the FactoryLink DDE Server.

Also, a FactoryLink application can use the FactoryLink DDE Client task set up through the Main Menu to initiate communication and retrieve data from external servers in products, such as Excel or Lotus.

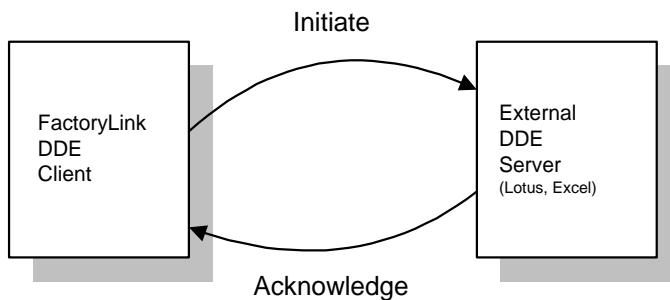
Either the server or the client can terminate DDE conversations. The following graphic illustrates this server/client relationship.

- **DYNAMIC DATA EXCHANGE (DDE) TASK DEFINITION**
- *DDE Conversations*
-
-

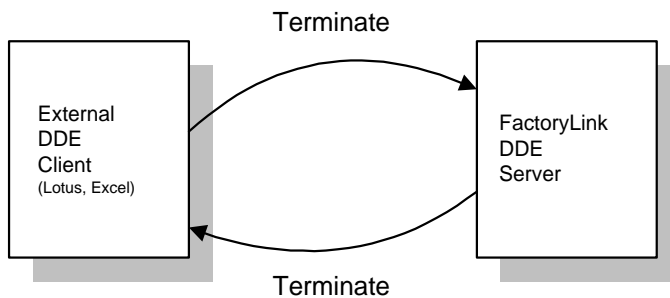
DDE clients access the FactoryLink DDE Server.



FactoryLink DDE Client can initiate DDE conversations with external DDE servers.



DDE conversations can be terminated by either the DDE server or the DDE client.



DDE Messages

The applications involved in a DDE conversation do not have to be specifically developed to work with each other since the applications communicate via DDE messages. DDE messages allow two or more Windows applications to communicate and share data.

FactoryLink supports the following standard DDE messages:

- **Advise:** Requests notification when an item changes
- **Execute:** Acknowledges the message
- **Initiate:** Start DDE conversation
- **Poke:** Send new value to the server
- **Request:** Request a value from the server
- **Unadvise:** Request that client is not notified of an item change
- **Data:** Notify client when an item changes
- **Ack:** ACK or NAK on message
- **Terminate:** Termination of the DDE Conversation

These messages are the basic building blocks the FactoryLink DDE Server and DDE Client work with.

- **DYNAMIC DATA EXCHANGE (DDE) TASK DEFINITION**
- *Setting Up DDE Server*
-
-

SETTING UP DDE SERVER

Perform the following steps to set up the FactoryLink DDE Server:

- 1 Complete the following fields to configure the DDE Server in the *System Configuration Information* panel:

Flag FSR in the Flag field. These flags allow the DDE Server to automatically load each time FactoryLink is started.

Task Name Name FLDDESRV to identify the Server task.

Path Path name as /bin/flddesrv.

Refer to Chapter 106, *“Using System Configuration”* for more information about the System Configuration Information panel.

- 2 Establish a DDE Link using the application-specific procedures for a DDE Formula or Create Link. Refer to the appropriate application documentation, such as the *Microsoft Excel User Guide*, for detailed instruction. DDE Links give Windows application access to the FactoryLink real-time database. For example, the value of a specified element can be set up to be displayed automatically in a cell of a Microsoft Excel spreadsheet.

DDE Server may be associated with either the SHARED or USER domain.

ESTABLISHING A DDE LINK

When a DDE client initiates a DDE conversation with the FactoryLink DDE Server using the INITIATE message, the client must specify a formula that consists of an Application, Topic, and Item.

- **Application**—The name of the server application. Use the values of the FactoryLink environmental variables **FLNAME.FLUSER**, separated by a period delimiter, to specify a FactoryLink application.

FLNAME is the environmental variable whose value is a FactoryLink application name. The default value of the environmental variable FLUSER is SHAREUSR. This default is established when the Run-Time Manager is started from the Start FactoryLink icon., Specify the FLUSER value used if the Run-Time Manager is started from the Run prompt.

- **Topic**— The name of the file containing the exchange data. For a FactoryLink application, use **RTDB** to represent the FactoryLink real-time database.
- **Item**— The actual data exchanged in a DDE conversation. This is the name of a specific FactoryLink tag name for a FactoryLink application.

Note: In Excel, array tag names must be enclosed in single quotation marks. These quotation marks are not used for non-array tag names.

Element names specified for DDE must be unique. Refer to “Determining Unique Elements” in the *FactoryLink Configuration Guide* for more information on determining a unique element.

The client must include the following string in its DDE message to obtain information from the FactoryLink real-time database:

```
=FLNAME.FLUSER|RTDB!'tag_name'
```

Note: If you are creating an application in a language other than English, check your country's system settings to verify the number format is set up properly. In addition, the DDE Item name can be country-dependent. For example, the English version of a DDE Item name called R1C1 becomes Z1S1 in German.

- **DYNAMIC DATA EXCHANGE (DDE) TASK DEFINITION**

- *Establishing a DDE Link*

-
-

Using a DDE Formula

Although DDE is implemented in different ways, the actual DDE commands are very similar.

The following example is specific to Microsoft Excel; however, the basic concepts can be applied to any product that supports DDE:

- 1 Open a spreadsheet in Excel.
- 2 Highlight the destination area. It must match the size and shape of the source information.
- 3 Type an equal sign.
- 4 Type the source application name. Use the values of the FactoryLink environmental variables FLNAME.FLUSER, separated by a period delimiter.
- 5 Type a vertical bar (|).
- 6 Type the source topic name (RTDB).
- 7 Type an exclamation point.
- 8 Type the address of the source information, which is any standard FactoryLink tag name enclosed in single quotes.

Using a Create Link Option

Many Windows applications use Create Link to establish a DDE Link.

The following example is specific to Lotus 123W; however, the basic concept can be applied to any DDE client supporting the Create Link.

- 1 Open the Edit | Link Options menu and choose Create Link. Enter the following information:
 - Type the value of the environmental variables FLNAME.FLUSER, separated by a period for application name.
 - Type RTDB for topic.
 - Type the name of any standard FactoryLink tag name For item name,.

The DDE Link is now in place and the client application can communicate with the FactoryLink real-time database.

MODIFYING A LINKED ELEMENT

Once a FactoryLink element is linked, it can be modified. The following example shows how an element linked to Excel is modified using an Excel macro.

Perform the following steps in Excel to create an Excel macro:

- 1 Choose File>New>Macro.
- 2 Open the appropriate spreadsheet in Excel.
- 3 Select cell A1.
- 4 Enter the following macro information in the spreadsheet.

Table 89-1 Spreadsheet Macro Information

| A | | B |
|---|---|---------------------------|
| 1 | SendChannel = INITIATE("FLNAME.FLUSER","RTDB") | Open a channel. |
| 2 | =POKE(SendChannel,"message1",B2) | Send this to FactoryLink. |
| 3 | =TERMINATE(SendChannel) | End DDE conversation. |
| 4 | =RETURN() | |

- 5 Choose Formula>Define Name.
- 6 Type a name for the macro and choose Command Type.
- 7 Enter =\$A\$1 in the Refers To field.
- 8 Open the Macro menu and choose Run.

- **DYNAMIC DATA EXCHANGE (DDE) TASK DEFINITION**
- *DDE Client Functions*
-
-

DDE CLIENT FUNCTIONS

Use the *DDE Client Table* to configure bidirectional communications between the FactoryLink real-time database and external Windows applications via a DDE link. This communication is established by entering data in configuration tables that enable the system to read data from and write data to an external DDE server.

The DDE Client task uses two types of configuration tables:

- **Read/Write Control**—This table allows the user to specify multiple DDE server configurations.
- **Read/Write Information**—This table allows the user to identify DDE item names to be read and transferred to the FactoryLink real-time database or real-time database values are written to.

The DDE Client task, through the DDE protocol, reads and writes data to and from a DDE server. Read and write operations can occur at timed intervals and upon triggered conditions, operator requests, or system events.

The DDE Client also times the response. If a time-out occurs or the response is not valid, the task determines the interface to be faulty and an error code is displayed on the Run-Time Manager screen indicating which device has the communication problem.

Read Operations

In a read operation, the DDE Client task performs a read of the DDE server item names specified in the *Read/Write table*.

The FactoryLink read operation can be of two types:

- **Block Read or Triggered**—Reads the values of the DDE server item names specified in a *Read/Write table* and writes them to the real-time database.

As part of the configuration of a *Read/Write table* in a triggered block read, the user specifies a trigger element in the real-time database to initiate read operations. When the value of this trigger element is 1 (ON) and has changed since the last scan, the DDE Client task performs a read of the DDE Server item names you specified in the *Read/Write table*.

Triggered block read operations allow the DDE Client task to achieve maximum performance by transmitting the minimum number of read commands necessary to collect the data specified in the *Read/Write table*.

- **Advise/Unadvise or Not Triggered**—The DDE Client task can be configured to notify the DDE server it wants to be updated when the values of the *Read/Write table* change. A digital tag value is associated with this feature. If the digital tag value is 1 (ON), the DDE Client alerts the DDE server it wants to be updated on changes of the *Read/Write table*. If the digital tag value is 0 (OFF), no such updating is performed.

Write Operations

In a write operation, the DDE Client task, through the DDE protocol, initiates a write operation to the DDE server item names specified by the user *Read/Write table*.

FactoryLink write operations can be of two types:

- **Block Write or Triggered**—Writes the values of all the elements specified in a *Read/Write table*.

In a triggered block write, as part of the configuration of a *Read/Write table*, the user specifies a trigger element in the real-time database to initiate write operations. When the value of this trigger element is 1 (ON) and has changed since the last scan, the DDE task performs a write to the DDE server item names you specified in the *Read/Write table*.

Triggered block write operations allow the DDE client to achieve maximum performance by transmitting the minimum number of write commands necessary to collect the data specified in the *Read/Write table*.

- **Exception Write or Not Triggered**—Writes only real-time database values that have changed or values of elements whose change-status flags have been set since the last time the task scanned the real-time database.

You can configure as many *Read/Write tables* as available memory allows, with the number of entries in each table also limited only by available memory.

- **DYNAMIC DATA EXCHANGE (DDE) TASK DEFINITION**
- *Setting Up DDE Client*
-
-

SETTING UP DDE CLIENT

Perform the following step to set up the FactoryLink DDE Client:

- 1 Configure the task in the *System Configuration Information* panel by specifying the following information in each field:

Flag **FSR** in the Flag field. These flags allow the DDE Client to automatically load each time FactoryLink is started.

Task Name Name **FLDDECLI** to identify the Client task.

Path Pathname as /bin/flddecli.

Refer to Chapter Running H/F 4, *Using System Configuration* for more information about other related System Configuration Information panel fields such as Status.

Event and Interval Timer Task Definition

Use the Event and Interval Timer task to define timed events and time intervals that initiate and control any system function in run-time mode.

- Timed events occur at a specific time not more than once every 24 hours (for example, Monday at 8:00 A.M.). They are configured in the Event Timer Table.
- Time intervals occur at least once every twenty-four hours at regular intervals of the system clock (for example, every 60 seconds). They are configured in the Interval Timer Table.

The Event and Interval Timer task links timed events and intervals to real-time database elements used as triggers whenever the event or interval occurs. It is defined in the SHARED domain.

The use of Event and Interval timers requires an understanding of change-status flags. Refer to the *FactoryLink Fundamentals* for this discussion.

There is no limit, except the amount of available memory, to the number of event and interval timers that can be defined.

This chapter introduces the operational concepts to configure Event and Interval Timer operations. Refer to “Event and Interval Timer” on page 37 in the *FactoryLink Configuration Guide* for information on procedures to configure this task. Use “Event and Interval Timer” on page 155 in the *FactoryLink Reference Guide* for a quick description of this task’s panels, its entries, and its associated error messages.

- **EVENT AND INTERVAL TIMER TASK DEFINITION**

- *Principles of Operation*

-
-

PRINCIPLES OF OPERATION

The Event and Interval Timer task operates in synchronization with the system clock. You must create a digital element in the real-time database for each defined interval or event. When the system clock matches the specified event or interval, the task forces the value of this digital element to 1 (ON).

The Event and Interval Timer task also updates global information used by FactoryLink, such as the current time, the day of the week, and the month. Such global information is stored in predefined FactoryLink real-time database elements, known as reserved elements. Each is one of the following data types: analog, long analog, or message.

The following table lists reserved elements the Event and Interval Timer task updates while the Timer task is running. These reserved elements are constantly updated. You must have entered an R flag for the Timer task in the System Configuration Table in the Configuration Manager Main Menu for the Timer task to run.

Table 90-1 Reserved Elements

| Reserved Element | Data Type |
|---|-------------|
| A_SEC | Analog |
| A_MIN | Analog |
| A_HOUR | Analog |
| A_DAY (Day of month) | Analog |
| A_MONTH | Analog |
| A_YEAR | Analog |
| A_DOW (Day of week) | Analog |
| A_DOY (Day of year) | Analog |
| DATE (DOW MM-DD-YYYY) | Message |
| TIME (HH:MM:SS) | Message |
| DATETIME (DOW MM-DD-YYYY HH:MM:SS) | Message |
| YYMMDD (YY-MM-DD) | Message |
| SECDAY (Seconds since start of current day) | Long Analog |
| SECYEAR (Seconds since start of current year) | Long Analog |
| SECTIME (Seconds since January 1, 1980) | Long Analog |

- **EVENT AND INTERVAL TIMER TASK DEFINITION**

- *Changing the Operating System Date and Time*

-
-

CHANGING THE OPERATING SYSTEM DATE AND TIME

Shut down the Timer task, change the date and time, and restart the task to change the operating system date and time while FactoryLink is running; otherwise, the Timer task tries to catch up by processing missed intervals.

File Manager Task Definition

The FactoryLink File Manager task can be used to perform basic operating system file-management operations initiated by a FactoryLink application at run time. The task can work in conjunction with FactoryLink's FLLAN option to initiate operations within other FactoryLink stations on a network. The File Manager initiates the following operations, which are performed by the operating system:

- Copy a file
- Rename a file
- Delete a file
- Print a file
- Display a directory
- Type a file

File Manager initiates these operations with six commands: COPY, REN, DEL, PRINT, DIR, and TYPE. These commands perform the same functions as their operating-system counterparts. The File Manager controls all file operations through the FactoryLink real-time database.

You can configure other FactoryLink tasks to initiate File Manager operations. These include the following:

- You can configure input functions in Graphics so an operator can use them to initiate file-management operations at run time, such as to display a list of recipes or reports.
- The Timer task can trigger File Manager to automatically back up files to a networked server at certain intervals, such as each day at midnight.
- The Timer task can also trigger File Manager to delete log files automatically at certain intervals, such as once every four hours, or after certain events, such as when log files reach a specified size.
- Alarm Supervisor can trigger File Manager to print alarm files.

- **FILE MANAGER TASK DEFINITION**
-
-
-

To enable a local node to perform file management operations with a remote node at run time, the FLFM_SERVER task must be running in the SHARED domain on the node that does not initiate the FLFM command. Either start the FLFM_SERVER task manually from the Run-Time Manager screen at startup or configure FactoryLink to start the FLFM_SERVER task automatically at system startup. Complete the following steps to do this.

- 1 Open the System Configuration Information panel in the SHARED domain from the Main Menu on the remote node.
- 2 Locate the row containing the entry FLFM_SERVER in the Task field.
- 3 Place the cursor over FLFM_SERVER.
- 4 Tab to the Flags field.
- 5 Enter an R in the Flags field. This configures the FLFM_SERVER task on the remote node to start up automatically whenever FactoryLink is started.

This chapter introduces the operational concepts to configure File Manager operations. Refer to “File Manager” on page 53 in the *FactoryLink Configuration Guide* for information on procedures to configure this task. Use “File Manager” on page 167 in the *FactoryLink Reference Guide* for a quick description of this task’s panels, its entries, and its associated error messages.

Historian Task Definition

The Historian task is the interface between FactoryLink and a relational database. It processes data requests from other FactoryLink tasks and sends them to the relational database. Data requests can store data in the relational database for Database or Data-Point Logging or retrieve data from the relational database for tasks like Trending or Database Browser.

FactoryLink communicates with many different Historians, depending on the relational database receiving the data request.

Configuration is predefined for the dBASE IV Historian and you do not have to configure an Historian if you are using Data-Point Logging. You need to know how to configure an Historian if you are using a different Historian or want to configure a Data-Point Logging table other than the ones provided with FactoryLink.

This chapter introduces the operational concepts to configure Historian operations. Refer to “Configuring the System to Work with a Historian” on page 247 in the *FactoryLink Configuration Guide* for information on procedures to configure this task. Use “Historian” on page 227 in the *FactoryLink Reference Guide* for a quick description of this task’s panels, its entries, and its associated error messages.

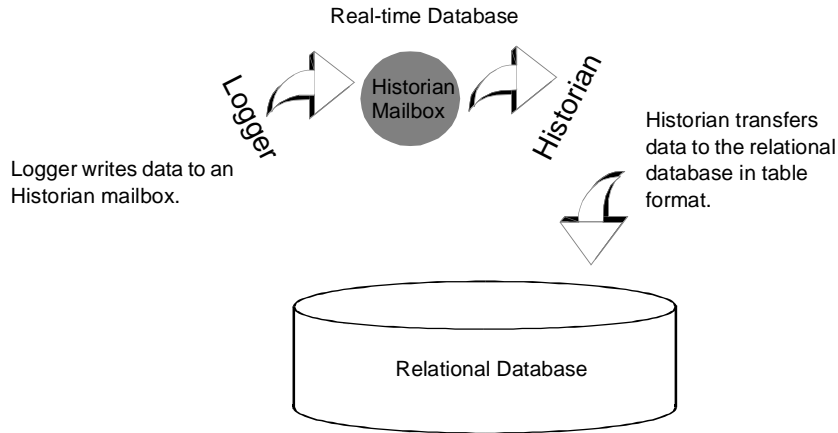
Historian Methodology

The following steps describe and illustrate how Historian processes data requests for a relational database:

1. A FactoryLink task sends a data request to a mailbox Historian service. This can be a request from Database or Data-Point Logging to store data in the relational database or from a task like Trending to retrieve data from the relational database. FactoryLink tasks submit their requests for data in the form of Structured Query Language (SQL) statements. In general, mailboxes are unidirectional; that is, a task requesting data from the Historian makes the request through a different mailbox than the mailbox Historian uses to return data.
2. Historian reads this mailbox and processes any queued data requests. It transmits the data request to the relational database server.

- **HISTORIAN TASK DEFINITION**
-
-
-

3. The relational database returns the requested information to the Historian if the request was to retrieve data.
4. Historian returns the requested data to the requesting task.



Supported Relational Databases

You receive dBASE IV Historian with FactoryLink. The dBASE IV Historian communicates with a dBASE IV relational database. If you use a different relational database in your application, you can configure an Historian that communicates with it. Each supported relational database has a different FactoryLink Historian. FactoryLink supports the following relational databases:

- dBASE IV
- Database2
- Informix
- Oracle7
- ODBC
- Sybase

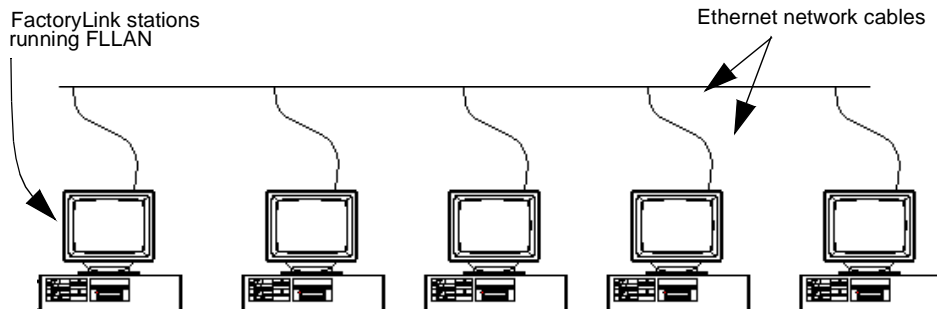
You must add the desired Historian to the system configuration as a task if you want to configure an Historian other than or in addition to the dBASE IV Historian.

Review the considerations and information at the beginning of each chapter about specific Historians in the Configuration manual before configuring a particular Historian.

Local Area Networking Task Definition

The FactoryLink Local Area Networking (FLLAN) module transmits FactoryLink data between computers or stations across a network. A network is a combination of hardware and software that lets multiple computers share resources, such as files, printers, or data. A network consists of the following parts:

- A Network Operating System (NOS)—Software that transports data between software applications on different computers.
- A network application—Software that sends data to a similar application on another computer via the Network Operating System. FactoryLink FLLAN is a network application. FLLAN can send data to multiple FactoryLink stations on a network, even if each station runs on a different operating system.
- The network hardware—Network interface cards installed on each computer on the network and cables that link them all together.



This chapter introduces the operational concepts to configure Local Area Networking operations. Refer to “Setting up the System” on page 617 in the *FactoryLink Configuration Guide* for information on procedures to configure this task. Use “FLLAN” on page 189 in the *FactoryLink Reference Guide* for a quick description of this task’s panels, its entries, and its associated error messages.

- **LOCAL AREA NETWORKING TASK DEFINITION**

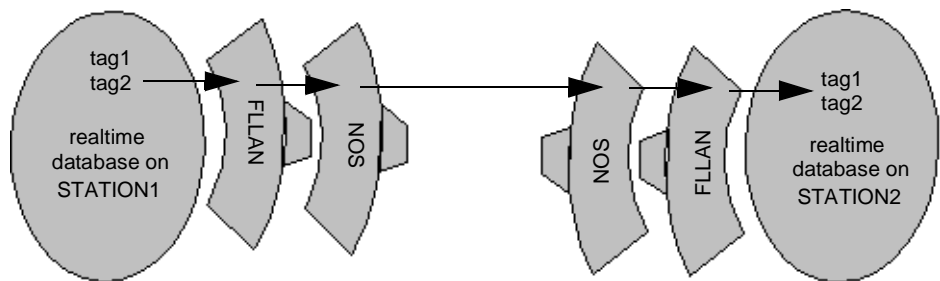
- *FLLAN Methodology*

-
-

FLLAN METHODOLOGY

The following steps describe and illustrate how FLLAN sends and receives data across a network:

1. FLLAN reads the data from the real-time database and passes it to the Network Operating System.
2. The Network Operating System translates the data into a form the other station understands before sending it across the network.
3. The Network Operating System on the receiving station receives the data and translates the data into a form FactoryLink understands before passing it to FLLAN.
4. FLLAN writes the data to the real-time database on the receiving station.



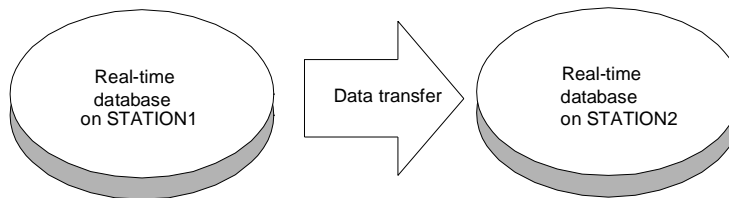
SENDING AND RECEIVING DATA

FLLAN uses send and receive operations to transfer data between stations on the network. This includes the following:

- Sending Values to Remote Stations
- Receiving Values from Remote Stations

Sending Values to Remote Stations

FLLAN uses send operations to send data to remote stations. During a send operation, the FLLAN on the local station sends values from the FactoryLink real-time database across the network to the FLLAN on the remote station. The FLLAN on the remote station writes these values to the real-time database on the receiving station.



FLLAN uses two types of send operations:

- **Block Send**—Sends the values of all elements configured as a block together. An event in FactoryLink triggered block sends. If data changes frequently and at regular intervals, send the data in triggered blocks because sending by exception can jam the network with data at irregular intervals. Block send is most efficient when the application sends a group of elements at one time to the remote station(s).
- **Exception Send**—Sends the values of elements that have changed since the last send. Use this method if the elements in the local station real-time database change infrequently. You minimize network traffic if you send only the values that change.

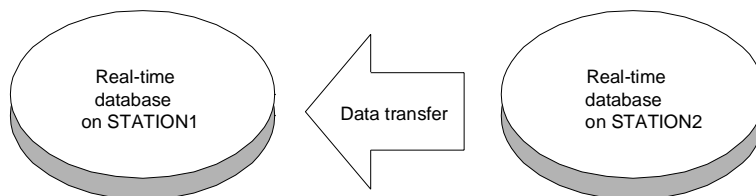
- **LOCAL AREA NETWORKING TASK DEFINITION**

- *Sending and Receiving Data*

-
-

Receiving Values from Remote Stations

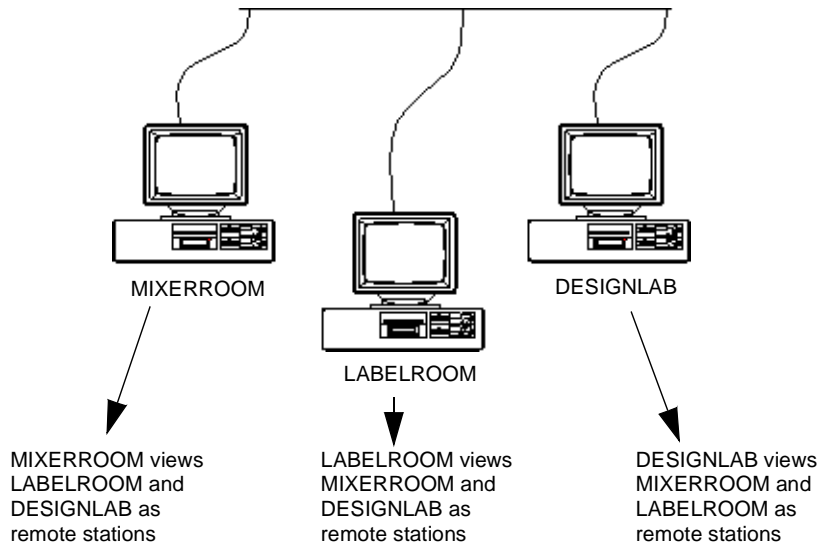
During a receive operation, FLLAN receives values from a remote station and stores them in the FactoryLink real-time database as elements.



You do not need the module FLLAN on two or more FactoryLink stations in order to share and store files on a network server or use network printers. External networking software allowing peer services is sufficient to achieve this goal.

LOCAL AND REMOTE STATIONS

Each station on the network must have a unique station name so FLLAN can identify it. Each station views itself as the local station and views all other stations on the network as remote.



For example, in a network with three stations, one station is assigned the name, MIXERROOM, another is assigned the name LABELROOM, and the third is assigned the name DESIGNLAB.

MIXERROOM views itself as local and views LABELROOM and DESIGNLAB as remote; LABELROOM views itself as local, and views MIXERROOM and DESIGNLAB as remote; and DESIGNLAB views itself as local and views LABELROOM and MIXERROOM as remote.

- **LOCAL AREA NETWORKING TASK DEFINITION**
- *Network Groups*
-
-

NETWORK GROUPS

You can combine one or more stations into groups. Grouping permits you to transmit the same data to multiple stations with a single operation. A single station can belong to more than one group.

You can use the same group name on more than one remote station; however, these groups are independent and do not correspond to each other.

For example, suppose you define a group named Paint on both MIXERROOM and LABELROOM.

- On MIXERROOM, Paint contains only one remote station, DESIGNLAB. When MIXERROOM sends data to Paint, it sends it to DESIGNLAB.
- On LABELROOM, Paint contains two remote stations, MIXERROOM and DESIGNLAB. When LABELROOM sends data to Paint, it sends it to MIXERROOM and DESIGNLAB.

USING MULTIPLE PLATFORMS ON A NETWORK

Because the Network Operating System is transparent to FactoryLink, you can use a different Network Operating System program on each station on a network. This lets you use FactoryLink for different platforms within the same network.

You must use the same protocol on all stations in the network.

Supported Protocols

FLLAN supports TCP/IP protocol.

Supported Network Operating Systems

The following tables list the network operating systems compatible with each protocol. A table exists for each platform supported by FactoryLink. Be sure to use the same network protocol on all stations on the network.

Table 93-1 FactoryLink for OS/2 Network Products

| Network Protocol | NOS Product |
|------------------|-----------------|
| TCP/IP | ftp PC TCP 1.3. |
| | IBM TCP/IP 2.0. |

FactoryLink for Windows NT and Windows 95 can communicate with any TCP/IP package that is WINSOCK-compliant. WINSOCK provides a unified front end to protocols. Refer to the TCP/IP vendor's documentation for details.

Table 93-2 FactoryLink for Windows NT or Windows 95 Network Products

| Network Protocol | NOS Product |
|------------------|---|
| TCP/IP | The TCP/IP provided with the Windows NT operating system software. This is WINSOCK-compliant. |

Table 93-3 FactoryLink for UNIX Network Products

| Network Protocol | NOS Product |
|------------------|---|
| TCP/IP | Any TCP/IP made by the same manufacturer as the UNIX operating system software. |

MONITORING THE NETWORK

You can monitor the status of remote stations on the network, such as the number of transmissions the remote station sent and received and whether these transmissions were successful. You can view the status at run time and other FactoryLink modules can use this information for other activities.

For example, Math & Logic and Alarm Supervisor can monitor these elements and trigger an alarm whenever a remote station disconnects.

You can monitor any FactoryLink station on the network running FLLAN.

- **LOCAL AREA NETWORKING TASK DEFINITION**

- *Local Station's Default Values*

-
-

LOCAL STATION'S DEFAULT VALUES

The local station name and the default values FLLAN uses to transmit data is stored in the local name file {FLAPP}/net/local on each FactoryLink station.

The following list includes the default values. Refer to “Naming the Stations and Network Groups” in the *FactoryLink Configuration Guide* for details on changing these values. We recommend you consult your network administrator if you need to change the default values.

- TX = 20
- RX = 60
- INIT = 0
- CALL = 10
- BUFSIZE = 512
- MAXLEN = 512
- MAXSESS = 32
- ACK = 0
- ST = 10
- SD = 10

TX (Transmit Timeout)

A number between 0 and 65527 that sets the maximum time in seconds between transmissions. The default is 20. If the local station does not send any data to a given remote station after the indicated time, the local station sends an I am still here packet to the remote station.

RX (Receive Timeout)

A number between 0 and 65527 that sets the maximum time in seconds between receptions. The default is 60. Ensure this value is at least three times greater than the TX value. If the local station does not receive any data from a remote station after the indicated time, the local station disconnects from the remote station and attempts to reconnect.

If you specify an RX value greater than 60, modify the `-t` program argument for Run-Time Manager in the System Configuration table; otherwise, FLLANRCV may not shut down properly. Refer to “Setting Up Program Arguments” on page 293 for more information about modifying program arguments.

Complete the following steps to do this:

- 1 Ensure the current domain selected is SHARED in the Configuration Manager Domain Selection box.
- 2 Choose System Configuration from the Configuration Manager Main Menu to open the System Configuration Information panel.
- 3 Choose FLLAN. The cursor highlights FLLAN SND and RCV.
- 4 Tab to the Program Arguments field.
- 5 Enter `-t` and the same value as RX. For example, if RX is 90, enter `-t90`.
- 6 Click on Enter to save the information.
- 7 Click on Exit to return to the Main Menu.

INIT

A value of 0 or 1 that specifies whether the local station sends all data when it first connects with another station. The default is 0. The local station uses this value only when a remote station starts up.

If you leave the value at 0, when the local station first connects to another station, it does not send values until one of the values has changed.

If you enter 1, when the local station first connects to another station, it sends all values during the first real-time database scan. This can be useful when you start up a remote station after the local station has been running. The new station has no values when it first starts up, so the local station sends all the values it has at that time. After that, the local station sends values as normal.

CALL

A number between 0 and 65527 that defines the minimum amount of time in seconds the local station waits for a call to a remote station to connect. The default is 10. If the remote station does not connect to the local station, the local station waits at least CALL seconds before attempting to reconnect. The remote station may still connect to the local station in the interim.

- **LOCAL AREA NETWORKING TASK DEFINITION**
- *Local Station's Default Values*
-
-

MAXLEN

The largest number of bytes a station can send or receive in a single data packet. The default is 512. This number can be from 512 through 1,048,576.

The tag data is truncated if a message or mailbox tag larger than MAXLEN is sent.

Make this number the same on all stations.

If you enter a value for this parameter less than the minimum, FLLAN uses the default minimum.

If you enter a value greater than the maximum, FLLAN uses the default maximum.

Only FLLAN uses this parameter. File Manager uses BUFSIZE to determine the buffer size.

Each element uses a specific number of bytes, depending on its data type. All elements use 4 bytes to store the tag name + *x* bytes to store the value, as shown in the following table.

Table 93-4 Bytes for Data Type

| The tag type... | uses... bytes | + 4 for the tag name | which = |
|-----------------|--|----------------------------------|---------|
| Digital | 2 | + 4 | 6 |
| Analog | 2 | + 4 | 6 |
| Longana | 4 | + 4 | 8 |
| Floating-point | 8 | + 4 | 12 |
| Message | the number of characters in the string | + 4 +2 bytes for the length | y |
| Mailbox | the number of characters in the string | + 4 + 26 bytes for the header | y |

The MAXLEN parameters in the LAN Local Names panel must be configured to specify the maximum number of bytes each node requires to send or receive a single data pack. MAXLEN parameters must match on all the nodes that receive data.

If you are using distributed alarms, use the following formula to calculate the number of bytes required at each node to distribute your alarms and logbook entries along the network.

$$((84 \times \text{active_alarms}) + 38) + (\text{number_of_logbooks} \times (24 + \text{msg_space})) = \text{bytes}$$

where

- | | |
|---------------------------|---|
| <i>active_alarms</i> | Maximum number of alarms defined for display in the Active Alarms field of the General Alarm Setup Control panel. |
| <i>number_of_logbooks</i> | Maximum number of logbooks expected to be generated for the alarms defined. This number can be smaller or equal to the <i>active_alarms</i> . A practical estimate of the normal volume of logbook entries is 20-30% of the total alarms. |
| <i>msg_space</i> | Number smaller or equal to the number of input lines. |

BUFSIZE

A number between 128 and 2048 that sets the size of each buffer in bytes. The default is 512 bytes. The size of the buffer determines the amount of data File Manager can transmit across the network in a single message.

Only File Manager uses this parameter. FLLAN uses MAXLEN to determine the buffer size.

MAXSESS

The maximum number of stations the local station can connect to at the same time. These are called connections. The default is 32. The maximum number of connections for the TCP/IP network protocol can be any number from 1 to 64.

ACK

A number from 0 to 1024 that specifies the number of seconds the local station waits for a remote station to send a data packet acknowledgment before disconnecting from that station. The default is 0, which indicates the local station does not require a data acknowledgment from a remote station.

- **LOCAL AREA NETWORKING TASK DEFINITION**

- *Local Station's Default Values*

-
-

ST (Send Timeout)

A number from 0 to 1024 that specifies the number of seconds the local station will keep trying to send its data if the remote station cannot accept it because it cannot process data fast enough. The default is 10 seconds. The local station generates an error when the timeout expires.

Caution: We recommend you not change this default.

SD (Send Delay)

A number from 0 to 1024 that specifies the number of milliseconds the local station waits between tries to send its data if the remote station cannot accept it because it cannot process data fast enough. The default is 10 milliseconds. If you increase this number, you will reduce CPU consumption but you may cause the overall performance to drop.

Caution: We recommend you not change this default.

ORDERING TAG NAMES

When sending element values, FLLAN groups the elements by data type and sends them in the following order: digital, analog, floating-point, message, long analog, and mailbox. These groups are called packets.

To maximize efficiency, place tags of like data-types next to each other in the LAN Send Information panel and order them by digital, analog, floating-point, message, long analog, and mailbox.

Example

If you enter the following tags in the panel:

ana2
flt2
ana1
msg1
flt1

FLLAN re-orders the tags, groups them, and sends the packets in the following order:

packet 1: ana2, ana1
packet 2: flt2, flt1
packet 3: msg1

Enter the same tags in the following order to maximize efficiency:

ana2
ana1
flt2
flt1
msg1

If you are concerned about the order within a data type, enter them in the order you want FLLAN to send them.

ana1
ana2
flt1
flt2
msg1

- **LOCAL AREA NETWORKING TASK DEFINITION**

- *Ordering Tag Names*

-
-

In versions of FactoryLink prior to 4.1.5, FLLAN sent the tags in reverse order. For example, if you entered tags in the following order:

ana1
ana2
flt1
flt2
msg1

FLLAN sent them in that order.

Math & Logic Task Definition

The FactoryLink Math & Logic task coordinates interactions among many of the FactoryLink tasks. These features include:

- Uses a structured, BASIC-like syntax
- Executes in the background with no operator intervention at run time
- Supports both interpreted and compiled Math & Logic (IML and CML)
- Supports block structures, looping constructs, and callable mathematical functions
- Supports automatic conversion of data types
- Supports multi-dimensional arrays

This chapter introduces the operational concepts to configure Math and Logic operations. Refer to “Configuring Math & Logic” on page 89 in the *FactoryLink Configuration Guide* for information on procedures to configure this task. Use “Math & Logic” on page 269 in the *FactoryLink Reference Guide* for a quick description of this task’s panels, its entries, and its associated error messages.

- **MATH & LOGIC TASK DEFINITION**

- *Uses*

-

-

USES

Math & Logic is called from other applications to perform functions such as:

- Manipulate real-time database elements and local variables
- Batch control
- Performing computational calculations
- Comparing logically two values stored in database elements
- Setting the value of an element to a value chosen by an end user at run time
- Controlling the system activity resulting from end-user interaction with on-screen buttons
- Validating end-user input
- Controlling screen changes
- Controlling information flow into and out of an application

PROCEDURES

Math & Logic executes procedures containing logically grouped and organized statements that have a single continuous thread of execution, or control flow, between the decision points (branches). Typically, you analyze the actions needed in the application and organize the operations needed for each action into one or more procedures. At run time, procedures can be either:

- Triggered in response to changes in the real-time database
- Executed directly from within another Math & Logic procedure

One or more procedures are contained within an editable plain-text program file. A program (.PRG) file can contain several separate procedures that may call (or refer to) one another.

CREATING PROGRAMS

You must create a program by configuring tables to use Math & Logic.

Configuration Tables

Use the Math & Logic configuration tables to:

- Define variables used by a task
- Identify the type of program, either interpreted or compiled
- Identify where the programs are stored
- Identify when a program should be invoked

The number of elements, triggers, and programs that can be defined is limited only by the amount of available memory, operating system, and/or compiler (in compiled mode only). Refer to the documentation supplied for compiler information.

You must configure the following three tables to create a program:

- **Math and Logic Variables table**—Defines Real-Time Database elements Math & Logic reads or writes to. An element must be defined before it can be used in a procedure or it will be considered an undefined variable and will not validate. Define each element referenced by the procedures in a program file by entering its tag name in this table. This table has one panel: Math and Logic Variables Information.
- **Math and Logic Triggers table**—Defines the elements used to trigger the Math & Logic procedures to run. For each trigger element defined, enter the associated procedure the trigger element is responsible for triggering and the method of execution: interpreted or compiled. This table has one panel: Math and Logic Triggers Information.
- **Math and Logic Procedure table**—Writes the Math & Logic procedures required by the application. Each program file must be named exactly as it is named in the Math and Logic Triggers Information panel but with a .PRG extension. After writing a procedure and saving it to a program file name, edit it in this panel. This table is a text-entry panel: Math and Logic Procedure - *filename.prg*.

- **MATH & LOGIC TASK DEFINITION**
- *Creating Programs*
-
-

Trigger Elements

Trigger elements used in the body of Math & Logic procedures must first be defined in the Math and Logic Variables table. Because many Math & Logic procedures can access them, these Real-Time Database elements are *global* in scope. Remember: global Real-Time Database elements are not the same as *global variables* which are declared inside program files and are used only within procedures. When a procedure is triggered by an element, the name of this element must be entered in the Math and Logic Triggers table so the procedure will load and run.

Completing the Tables

Complete the configuration tables in the following order:

- Math and Logic Variables table
- Math and Logic Triggers table
- Math and Logic Procedures table

Converting Previous Versions

Previous versions of Math & Logic operated in the IML mode only. Run the FLCONV utility to convert previous versions to the new one.

Opening the Configuration Manager Main Menu

Refer to *FactoryLink Fundamentals* for information on how to open the Configuration Manager Main Menu screen.

MODES

Math & Logic runs in one of two modes:

- Interpreted mode
- Compiled mode

Interpreted Mode

Interpreted mode is a subset of Compiled Math & Logic. IML means when the values of trigger elements associated with one or more procedures change in the real-time database, IML determines which procedures are affected. Math & Logic then interprets every line of the instructions and executes them for each triggered change. IML is excellent for application prototyping and logic debugging.

Most applications written in the Interpreted Mode function can be used with limited or no modifications under the Compiled Mode. If an application running in Interpreted Mode uses any reserved words as variables or procedure names, these will need to be modified so the compiler will be able to compile and link the procedures error free. These words include any reserved by the C compiler for the platform you are running on as well as those reserved by FactoryLink.

Using IML

When you use IML, you create procedure files that are scripts and manipulate both real-time database elements and local variables.

Interpreted mode is most appropriate when:

- Only mathematical functions will be performed
- A compiler is not available

Running IML

After starting the application,

- Load the program into memory
- Validate the program
- Wait for changes to the trigger elements in the real-time database associated with the procedures in the program
- Execute the program associated with the trigger when the trigger element is set to 1 (ON)

- **MATH & LOGIC TASK DEFINITION**
- *Modes*
-
-

Executing IML

When procedure changes are triggered in the real-time database, IML determines the proper procedure requested and then interprets and executes the instructions in the procedure.

Each time an IML program is executed, Math & Logic first reads, or interprets, the instructions within the program to determine the actions to perform. Then, it executes those actions.

Switching from IML to CML

Use the following procedure to make switching from IML to CML in large applications a one-step process:

Issue the following command from the BH_SQL prompt in the FLAPP directory of the application:

- To change from IML to CML, use the following command:

```
SQL>update imltrim set mode = 'COMPILED' where mode is not null
```

- To change from CML to IML, use the following command:

```
SQL>update imltrim set mode = 'INTERPRETED' where mode is not null
```

Compiled Mode

Compiled mode is a combination of several FactoryLink utilities with a third party ANSI C language compiler working together to generate ANSI C code from user created *.prg files.

In CML, the procedure program file is translated into C source code that is compiled to create a binary executable file for the platform the application is developed on. Because the application consists of both the SHARED and USER domains, CML creates two executable files, one for each domain. These executables are unique and are named:

- For SHARED domain:/{FLAPP}/SHARED/CML/CSHARED.EXE
- For USER domain:/{FLAPP}/USER/CML/CUSER.EXE

These files perform the same actions described in the program file when the associated trigger elements are set. CML increases the speed and functionality of applications that use IML.

Using CML

When you use CML, you make improvements on applications that use complex loops, such as nested IF-THEN-ELSE clauses or WHILE loops. Interpreted mode handles the execution of arithmetic operations fairly quickly, but CML adds more functionality for you as an in-line function to pass simple parameters and as an embedded C code block to pass complicated parameters.

Compiled mode is most appropriate when you want

- Faster startup and execution for improved performance
- Better handling of complex, multiple conditional statements
- Ability to call C functions or to insert C code into procedures
- Availability of extensive flow control for the application

Running CML

After starting the application:

- Wait for changes to the trigger elements in the real-time database associated with the procedures in the program
- Execute the program associated with the trigger when the trigger element is set to 1 (ON)

Executing CML

Each time a compiled program is executed, CML:

- Reads or interprets the instructions within the program to determine the actions to perform
- Executes these actions

CML Operation

CML uses three utilities for specific roles in creating the executables used at run time. These utilities are

- MKCML
- PARSECML
- CCCML

- **MATH & LOGIC TASK DEFINITION**

- *Modes*

-
-

MKCML

The MKCML utility is a shell that calls the PARSECML and CCCML utilities as needed for the current application. For each domain, MKCML checks the dependencies between the configuration tables and the program files by

- Ensuring the IML.CT file is up to date by calling CTGEN. CTGEN compares IML.CT against the database files upon which IML.CT is dependent. If the database files have a later time/date stamp than IML.CT, CTGEN rebuilds IML.CT to bring them up to date.
- Checking the time/date of IML.CT to determine if the Math & Logic configuration has changed. If so, it reproduces and recompiles all of the .C files by calling PARSECML and CCCML.

When you redirect the output of MKCML to a file, the messages in the dump may appear out of order because of the way the operating system buffers and outputs messages when redirecting output. If you do not redirect the output of MKCML, the MKCML reports the messages to the standard output in the correct order.

PARSECML

The PARSECML utility parses the application program file and produces .C files for a given domain. It produces a .C file for each program file listed as compiled (has COMPILED in the Mode field) in the Math and Logic Triggers Information panel.

The utility also checks the dependencies between the program field and the .C files to determine if any procedures have been updated since the .C files were last produced.

TRIGGERING/CALLING

Triggering

Developing Triggering Schemes

Multiple procedures are executed using the triggering method of this system. To develop the most efficient triggering scheme, consider the following:

- What intervals or circumstances trigger individual procedures at a reasonable rate for this data?
- Try to use conditional event triggering based on changing data so procedures run when data is change-state driven or trigger driven.

Single-threaded Task

This task is single-threaded. This means triggered procedures are cued and executed in the order in which they are triggered. A procedure that does not complete because of an infinite loop, for example, will stop the rest of the procedures from executing.

When running one procedure, set a trigger for the next.

Calling

You can both trigger and call procedures.

Trigger Procedure—If you want to wait until one procedure has completed before executing another or if your procedure has to execute when an event occurs, trigger a new procedure.

Calling Procedure—If you want a procedure to occur immediately, you must call the procedure.

This system uses a complex cueing process that necessitates using calling procedures if you want them to occur in a distinct order.

- **MATH & LOGIC TASK DEFINITION**

- *Triggering/Calling*

-
-

- Completing the Calling Sequence**

- Enter FLRUN command
 - FLRUN calls the MKCML utility
 - MKCML calls PARSECML to produce .C files (C code) from the program files
 - MKCML then calls CCCML to compile the .C files into object files using an external compiler and to link the object files into binary executables using an object linker.

OLE Automation Server Task Definition

This chapter explains how to use the FactoryLink OLE Automation Server (FLOLE). Refer to the Microsoft OLE documentation for a definition of how an OLE container application can directly open the OLE object functions. OLE automation is available in many products, including:

- Visual Basic
- Visual C++
- Borland C++

This FactoryLink task provides the necessary object classes to allow these OLE-compliant applications to directly read and write elements in the FactoryLink Real-time Database (RTDB). OLE Automation provides this ability. The OLE Automation Server chapter contains information about the following topics:

- “Principles of Operation”
- “Configuring the OLE Automation Server”
- “FactoryLink 6.5.0 Tag Data (FLOCX)”
- “FactoryLink ODBC Data Source”
- “Configuration of DCOM”
- “Windows NT Accounts”
- “Windows NT Distributed COM Configuration”
- “Visual Basic Usage”
- “Error Messages”

This chapter introduces the operational concepts to configure OLE Automation Server operations. Refer to “OLE Automation Server” on page 689 in the *FactoryLink Configuration Guide* for information on procedures to configure this task.

- **OLE AUTOMATION SERVER TASK DEFINITION**

- *Principles of Operation*

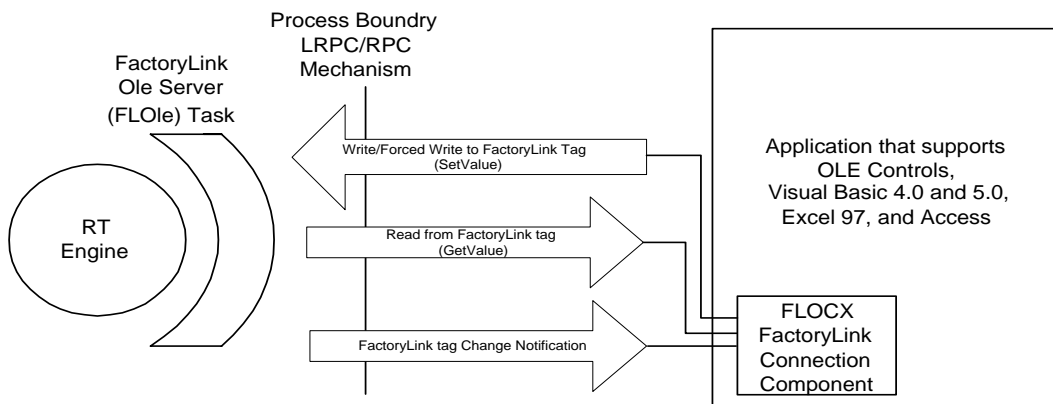
-
-

PRINCIPLES OF OPERATION

The FactoryLink OLE Server task provides other applications access to the FactoryLink real-time engine. Any application that supports OLE custom controls can interface with the OLE Server task using the FactoryLink Connection Component (FLOCX).

The FLOCX is an ActiveX control designed to interface with the OLE Server task. ActiveX is a technology that enables software components to interact with one another in a networked environment regardless of the language they were created in. ActiveX controls are reusable software components developed to quickly add specialized functionality desktop applications and development tools.

The diagram below shows an overview of the relationships between the FactoryLink real-time engine, the FactoryLink OLE Server task, the FLOCX, and the container application.



Persistence Task Definition

Use the Persistence task to save the values of an active FactoryLink application at predetermined times so if FactoryLink shuts down unexpectedly, useful data is not lost. When you restart FactoryLink with the warm start command-line option, the Run-Time Manager restores the last save of the real-time database from the persistence save file.

PERSISTENCE OVERVIEW

The memory-based real-time database represents the current state (values) of elements. The real-time database is a collection of domain instances. Elements in the database may have default values that are placed into the domain instance when it is initialized.

The values of the elements in a domain instance are lost when the domain instance is closed. When the domain instance is opened again, its elements are initialized to their default values.

This can be a problem if FactoryLink unexpectedly shuts down because of an event, such as a power loss or a faulty process. Useful information can be lost because of the initialization of the real-time database to its default values when the system is restarted. Since most of the information in FactoryLink applications is the accumulation of data over time, you cannot recover this data unless it is saved. Persistence provides a way of saving the state of an active FactoryLink application.

Persistence is the ability of an element to maintain its value over an indefinite period of time. Non-persistent elements lose their value when the Run-Time Manager exits and shuts down the real-time database; however, the values of persistent elements are written to disk so they are not affected when the real-time database shuts down.

The job of the Persistence task, therefore, is to save persistent data. The task offers the following features:

- Runs on all FactoryLink platforms.
- Lets you configure on a per-tag basis or on entire domains which elements are persistent and when their values are saved to disk.

- **PERSISTENCE TASK DEFINITION**

- *Persistence Overview*

-
-

- Saves one or all domain instances to disk while a FactoryLink application is running.
- Allows you to specify either a warm or cold start when initializing a domain instance.

Cold start initializes all elements to their default values as configured in the Configuration Manager Main Menu. Persistent elements are not restored to their previous values but are initialized to their default values.

Warm start initializes all non-persistent elements to their default values just like a cold start and restores all persistent elements in the domain instance to their previously saved values.

- Uses its own internal disk cache to increase speed and reduce disk I/O overhead.
- Resolves configuration changes to the application.
- Allows you to specify a trigger element that triggers the Persistence task to copy the current save file to a backup file.

This chapter introduces the operational concepts to configure Persistence operations. Refer to “Persistence” on page 21 in the *FactoryLink Configuration Guide* for information on procedures to configure this task. Use “Persistence” on page 295 in the *FactoryLink Reference Guide* for a quick description of this task’s panels, its entries, and its associated error messages.

PRINCIPLES OF OPERATION

Persistence begins during application development. Whether configuring a new application or reconfiguring an existing application, you must first determine which elements are persistent, when the values of the persistent elements are saved to disk, and how these saved values are restored during a warm start. Then, specify this information on the Tag Definition dialog when defining a persistent element.

At run time, the Persistence task saves the values of the persistent elements to its own internal disk cache and the task writes the data to disk from there. Saving the persistent values to memory first increases processing speed and ensures all values meant to be saved are saved within the allotted time.

The Persistence task runs under each domain that requires persistent data to be saved. The RESOLVE program executed by the FLRUN command creates a blank persistence save file for each domain the first time it is executed. During its initialization, the Persistence task loads the persistence save file to determine which elements in the application are persistent and when the values of those elements are to be saved. It also loads the PERSIST.CT file to get specific information about the configuration of the Persistence task itself.

When you perform a warm start, the current domain's Run-Time Manager restores the domain instance's real-time database from the persistence save file. It restores the values of the persistent elements to the last values saved by Persistence.

Use the warm start argument `-w` to perform a warm start of FactoryLink. In Windows and OS/2, add the `-w` to the command line for the icon used to start FactoryLink; in UNIX, either pass the `-w` to the FLRUN command or add it to the line in the script file used to start FactoryLink.

- **PERSISTENCE TASK DEFINITION**
- *Principles of Operation*
-
-

See the following table for details.

Table 96-1 Warm Starts Using -W

| Starting FactoryLink using the -w (warm start) | |
|---|--|
| Windows | <p>Create a new Start FactoryLink icon following the steps below:</p> <ol style="list-style-type: none"> 1. Click once on the Start FactoryLink icon to select it. 2. Click Edit>Copy. 3. Click Edit>Paste. A copy of the Start FactoryLink icon is pasted into the FactoryLink Program Group. It is labelled Copy of Start FactoryLink. 4. Change its properties by: <ol style="list-style-type: none"> a.) Clicking File>Properties b.) Clicking in the command line c.) Pressing the End key to get to the end of the command line d.) Adding -w to the end of the command line preceded by a space. 5. Click File>Rename and change the icon label to Warm Start FactoryLink. Press the Enter key. |
| OS/2 | <p>Create a new Run Time Manager icon following the steps below:</p> <ol style="list-style-type: none"> 1. Right click once on the Run Time Manager icon to select it. 2. Click Copy. <ol style="list-style-type: none"> a. Enter Warm Start for the icon new name. b. Click Copy. 3. Right click once on the Warm Start icon to select it. 4. Click Settings. In Optional Parameters, type -w. 5. Close window. |
| UNIX | <p>Edit the command line as:</p> <pre>\$ flrun -w -d -nshared <ret></pre> <p>or add the -w to the script file using the syntax above.</p> |

The FLRUN command executes the RESOLVE program before starting the Run-Time Manager to check the persistence save file for any configuration changes you made. The need to check for configuration changes is discussed in “Resolving Configuration Changes” on page 147.

RESOLVING CONFIGURATION CHANGES

After you shut down a FactoryLink application, you might reconfigure part of the application using the Main Menu or the Application Editor. This means the elements and their values stored in the persistence save file either may not exist or might not have the same data type when you restart the application. Before each FactoryLink session is restarted, the element names stored in the persistence save file must be checked for changes against the OBJECT.CT and the DOMAIN.CT files.

The RESOLVE.EXE program (resolve on UNIX systems) resolves the configuration changes. The FLRUN command automatically executes this program before it starts the Run-Time Manager for a particular FactoryLink session.

The RESOLVE program serves three purposes:

- Creates the blank persistence save file the first time it is run
- Manages the changes between the persistence save file and the FactoryLink configuration files
- Determines if the persistence save file is usable and, if not, the program looks for and uses the persistence backup file

RESOLVE makes the following changes:

- Removes element names from the persistence save file that have been deleted from the application or changed to a different data type
- Updates the element name ID for element names that were deleted, then recreated
- Adds element names to the persistence save file that have been reconfigured to have persistence (these element names are added with no data values.)
- Copies the persistence backup file over the persistence save file if the save file is corrupted

- **PERSISTENCE TASK DEFINITION**
- *Resolving Configuration Changes*
-
-

PowerNet Task Definition

Data collected or computed by FactoryLink is stored in data elements in the real-time database. Tag names defined in the Application Editor or in Configuration Manager panels identify these elements. Real-time database elements can be shared among FactoryLink applications running on the same or a different workstation (node) using PowerNet.

One FactoryLink application can act as a client/server. This application can serve other FactoryLink applications by providing needed information. As a client, the application can use information provided by other FactoryLink applications.

On platforms where FactoryLink supports multiple applications running on the same computer, you can run multiple instances of PowerNet. See *FactoryLink Fundamentals* for a discussion of multiple-user environments.

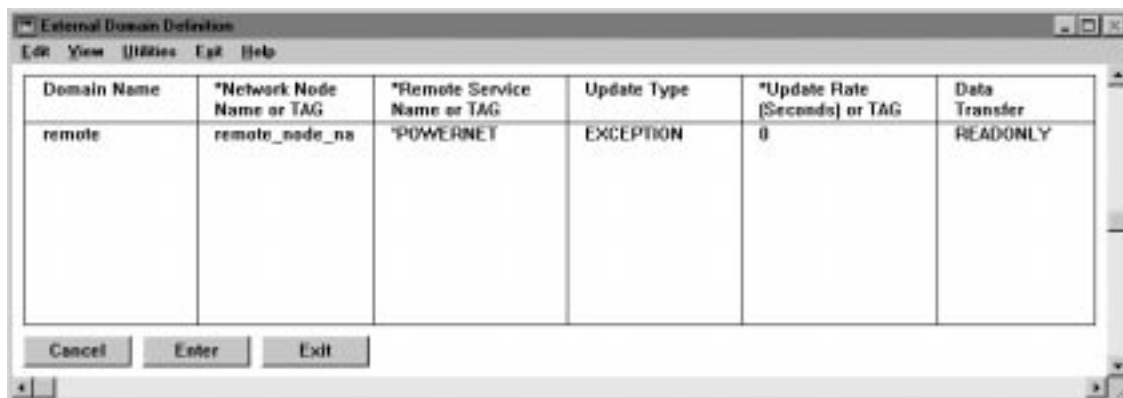
This chapter introduces PowerNet and associated terms, definitions, and concepts.

This chapter introduces the operational concepts to configure PowerNet operations. Refer to “Configuring Network Software” on page 593 in the *FactoryLink Configuration Guide* for information on procedures to configure this task. Use “PowerNet” on page 303 in the *FactoryLink Reference Guide* for a quick description of this task’s panels, its entries, and its associated error messages.

- **POWERNET TASK DEFINITION**
- *Using PowerNet with the FLDEMO Application*
-
-

USING POWERNET WITH THE FLDEMO APPLICATION

The following entries must be made in the External Domain Definition panel in the Configuration Manager to use PowerNet with the FLDEMO application provided with FactoryLink.



In the FLDEMO application, the PowerNet task is not automatically started. Start PowerNet from the Run Manager screen or add the R flag to the task in the System Configuration panel. See “System Configuration Information Panel” in the *FactoryLink Configuration Guide* for information about adding the R flag.

DEFINITIONS

Client/server refers to a distributed processing model where the client object initiates connections to and requests data from or actions of the server object. The server object, in turn, accepts or rejects such requests. A single PowerNet task can act as a client, a server, or both.

Client application refers to a FactoryLink application that references tags in a different FactoryLink application.

Server application is a FactoryLink application that sends data to a requesting client application.

Exception or unsolicited data refers to one or more FactoryLink data objects processed only when their data attributes indicates a changed state (the data value(s) may or may not have actually changed). In this case, the server object transmits unsolicited data to interested clients only (1) when the client has indicated interest in the data and (2) when changes are indicated for the data.

Polled or solicited data refers to one or more FactoryLink data objects processed only on-demand. In this case, a client object must send a request to a server object before the server will transmit the requested data.

Polled-exception data refers to one or more FactoryLink data objects whose processing is only considered on an on-demand basis but is actually limited to the subset of objects whose attributes indicate a changed state at the time of consideration. In this case, a client object must send a request to a server object, and the server will transmit only the subset of data that has changed since last transmission. This is also referred to as smart polling.

External domain refers to a named, logical grouping of FactoryLink data elements or tags whose owning application is not the local application. External domains have properties, such as network node and service names, update method (exception data or polled-exception data), and update frequency. If required, any number of external domain names in a given application may reference a single remote application, each having different update methods and rates. Each external domain defines a connection from the application to a server application.

Remote TAG (object) reference refers to a FactoryLink data element or tag whose name implies (but does not require) the element exists in an external domain. A tagname prefixed with an external domain name and a colon is a remote reference.

- **POWERNET TASK DEFINITION**

- *PowerNet Methodology*

-
-

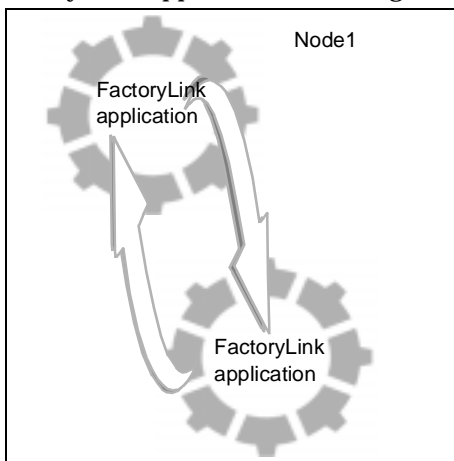
POWERNET METHODOLOGY

The following steps describe and illustrate how data stored in one FactoryLink application (server) is referenced by another FactoryLink application (client):

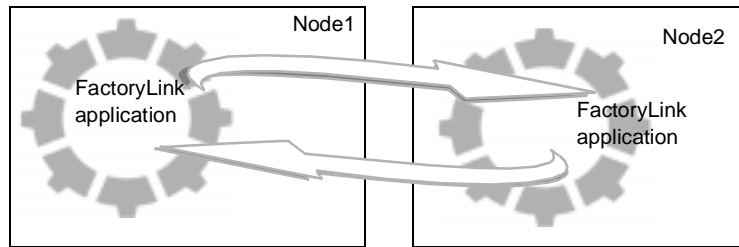
1. The client application requests all remote tag references on connection to a server.
2. A server transfers the data to the FactoryLink client application according to a client-defined schedule. This can either be on exception or on a polled interval.
3. Optionally, when data changes in the real-time database of the FactoryLink client application, the client application writes data back to the FactoryLink server application.

A distributed FactoryLink system can share data in any one of the following configurations:

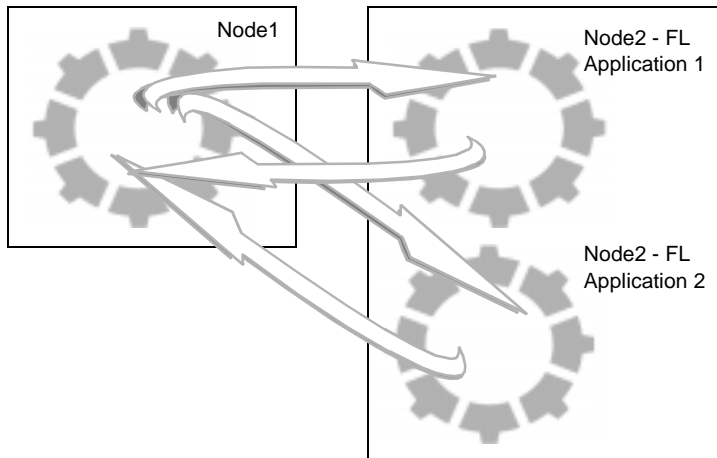
- Between two FactoryLink applications running on the same node.



- Between two FactoryLink applications running on two different nodes.



- Between multiple FactoryLink applications on different nodes.

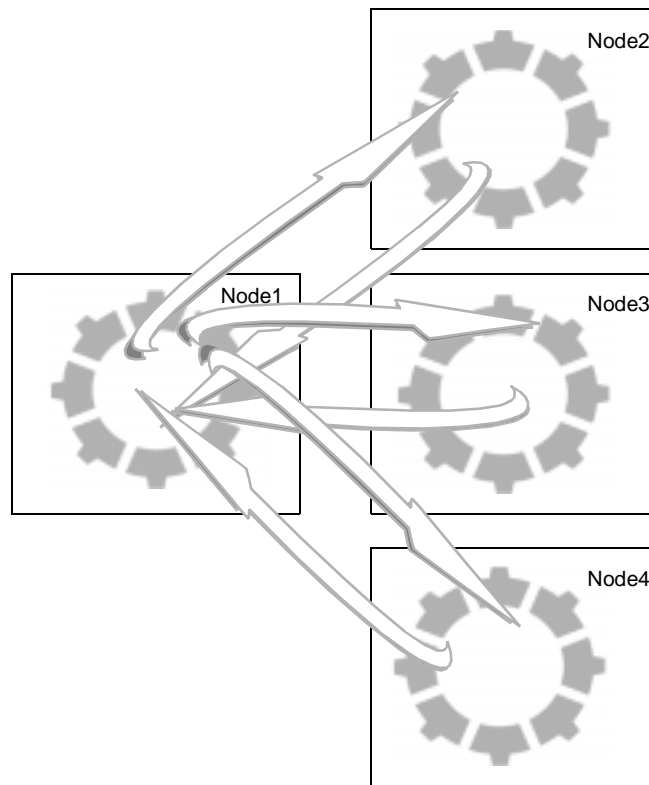


- **POWERNET TASK DEFINITION**

- *PowerNet Methodology*

-
-

- Between one application on one node and applications on different nodes.



You can configure the application to permit each of these connections to be changed at run time. This permits the operator at a client application to change the node being used as its server.

METHOD OF DATA TRANSFER

This section describes what initiates data transfer between a server application and a client application.

Startup

As each client attaches to a server application, all of the data shared between the server and client applications is transmitted from the server to the client. This ensures the client contains up-to-date data immediately upon starting up. This also occurs at reconnection in the event a connection is lost between the client and server.

Server-to-Client Data Transfer

Data transfer from the server to the client is configured by one of the following two methods:

- **Exception Data**—Transmits data to the client only when data has changed in the server application. This is the most common method of data transfer to a client and, depending on the stability of data on the server node, limits the network bandwidth usage.
- **Polled Data**—Transmits data to the client on a fixed interval, a dynamic interval, or at any event the client application generates. In order to reduce the total bandwidth utilization, PowerNet utilizes smart polling whereby only the data that has changed since the last poll is transferred to the client task at the polled event.

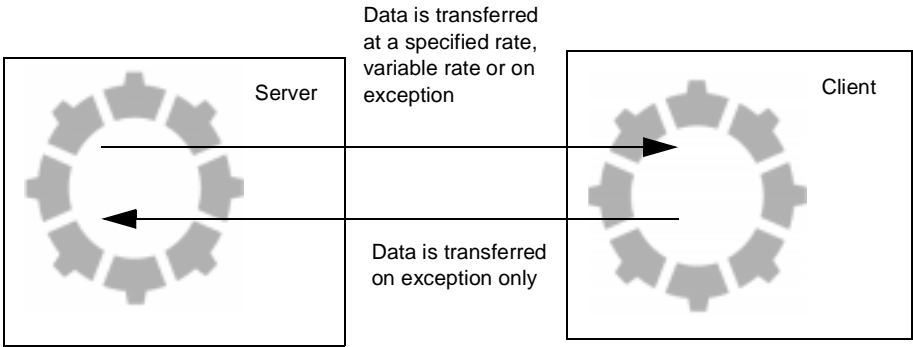
The polling interval can be specified as either a constant (for example, 10 seconds) or as a tag, allowing dynamic change to the interval at run time.

- **POWERNET TASK DEFINITION**
- *Method of Data Transfer*
-
-

Client-to-Server Data Transfer

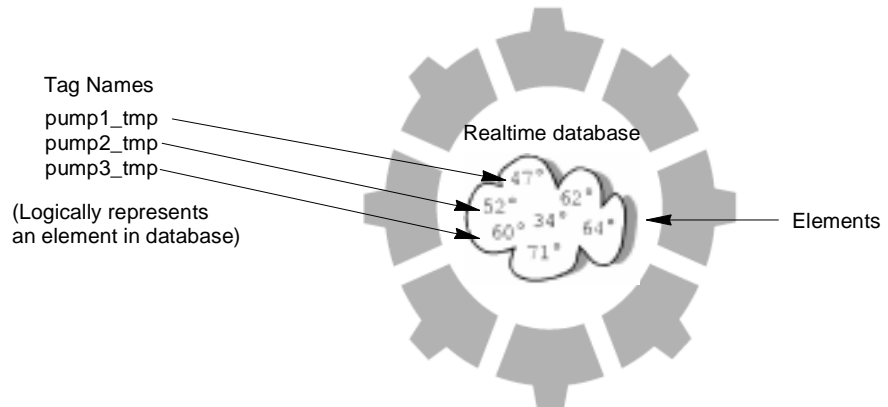
A data connection to an external domain may be configured as read only (the default) or read/write. Read/write connections allow data that has changed in the client application to be written back to the server application.

The diagram below illustrates supported data transfer for a simple client/server configuration.



DEFINING TAGS

Data stored in the real-time database is an element. Each element is assigned a logical name called a tag. This tag is used to reference the element in the real-time database.



Once an element is defined, you can make unlimited references to this element. Any FactoryLink task containing a reference to an element can read and write data to and from the element at run time.

During development, FactoryLink stores tag names in the FLAPP directory in the object database table. This information is updated to the .CT files when the run-time application is started.

Some tags are already defined in FactoryLink when it is shipped. Others are defined during application development either within the Configuration Manager or the Application Editor.

This chapter describes how to define tag names for database elements and provides some suggestions on how to use tags in your application.

TAG TYPES

PowerNet can transfer any tag type including mailboxes. However, PowerNet does not route mailbox replies like FLLAN does. It can only send mailbox tags that don't require a reply.

- **POWERNET TASK DEFINITION**

- *Tag Naming Guidelines*

-
-

TAG NAMING GUIDELINES

The following guidelines apply when you assign a tag name:

- 1-32 characters long

If using an array, add delimiters of up to 16 characters.

Refer to “Defining Element Arrays” on page 207 for more details.

If using PowerNet, use the model:

32 characters less 7 characters = 25 characters.

If using Scaling and Deadbanding, use the model.

32 characters less 7 characters less 9 characters.

Refer to the *Application Editor Guide* for details on the added extensions.

- Valid characters are A-Z, 0-9, _, @, \$, ., :
- Do not start with a number
- No embedded spaces

System Added Extensions

When defining tag attributes in the Tag Definition dialog, new tags are created automatically from the originally defined tag name if you are using PowerNet or Scaling and Deadbanding.

PowerNet

When a new tag is created, an extension of up to 6 characters plus a dot (7 characters maximum) is added. These additional characters reduce the maximum length of the original tag name; therefore, the maximum effective length of the original tag name is 32 less 7, or 25. If the original tag name plus extension exceeds 32 characters, a warning is issued.

Scaling and Deadbanding

When a new tag is created, an extension of up to 8 characters plus a colon (9 characters maximum) is added. These additional characters reduce the maximum length of the original tag name; therefore, the maximum effective length of the original tag name is 32 less 7 less 9, or 16.

If you are not sure if you will be using PowerNet or Scaling and Deadbanding, you may choose to define tag names using only 16 characters. Also, remember if shortening tag names, do not reduce the length of the extensions.

TAG TYPE CONVERSION

Tags that reference data in a remote application can be of a different data type from the tag definition in the remote application. The following chart identifies the supported data type conversions.

Table 97-1 Data Type Conversions

| Local Application | Remote Application |
|-------------------|--|
| Digital | Digital, analog, longana, float, message |
| Analog | Digital, analog, longana, float, message |
| Longana | Digital, analog, longana, float, message |
| Float | Digital, analog, longana, float, message |
| Message | Digital, analog, longana, float, message |
| Mailbox | Mailbox |

Precision loss should be considered in mixing data source types. In mixed data type assignments, the precision of the element with the least number of significant digits is the assumed precision. This should also be considered in transferring data between platforms that have different precision for like tag types.

Use caution when implementing tag type conversion. For instance, a message tag on the server application referenced as an analog tag in the client application must contain only numeric characters or the conversion will be to a value of zero. Additionally, an analog tag in the server application referenced as a digital in the client application will be assigned a value of 1 when the analog value is non-zero and 0 when the analog value is zero.

- **POWERNET TASK DEFINITION**
- *Tag Type Conversion*
-
-

Print Spooler Task Definition

The FactoryLink Print Spooler task permits you to direct data to printers or other devices with parallel interfaces and also to disk files.

The Print Spooler task also provides other features:

- File name spooling (loads file when print device is available, minimizing required memory)
- Management of printing and scheduling functions

Print Spooler receives output from other FactoryLink tasks, such as Alarm Supervisor or File Manager, and sends this output to a printer or disk file.

You can define up to five devices with Print Spooler to receive output from other FactoryLink tasks. FactoryLink tasks reference the corresponding device number in a configuration table to send files to one of these devices.

This chapter introduces the operational concepts to configure Print Spooler operations. Refer to “Print Spooler” on page 83 in the *FactoryLink Configuration Guide* for information on procedures to configure this task. Use “Print Spooler” on page 449 in the *FactoryLink Reference Guide* for a quick description of this task’s panels, its entries, and its associated error messages.

- **PRINT SPOOLER TASK DEFINITION**
-
-
-

Programmable Counters Task Definition

Use the Programmable Counters task to provide count-per-unit-of-time measurements and to provide event delays, such as defining a trigger to unlock a door and then specifying a delay before the door locks again.

A programmable counter is a group of elements with values that work together to perform a count. Outputs from programmable counters can be used to provide input to Math & Logic programs or other FactoryLink tasks or to trigger Math & Logic programs.

There is no limit, except the amount of memory, to the number of programmable counters that can be defined.

The Programmable Counters task uses either the SHARED or USER domain.

The use of programmable counters requires an understanding of change-status flags. Refer to *FactoryLink Fundamentals* for this discussion.

This chapter introduces the operational concepts to configure Programmable Counters operations. Refer to “Programmable Counters” on page 45 in the *FactoryLink Configuration Guide* for information on procedures to configure this task. Use “Programmable Counters” on page 459 in the *FactoryLink Reference Guide* for a quick description of this task’s panels, its entries, and its associated error messages.

- **PROGRAMMABLE COUNTERS TASK DEFINITION**

- *Principles of Operation*

-
-

PRINCIPLES OF OPERATION

A programmable counter is a group of elements with components that work together to perform a count. Each programmable counter is made up of some or all of the following elements and analog and digital values.

Elements

- Enable—triggers counting activity
- Up Clock—initiates the count upward
- Down Clock—initiates the count downward
- Clear—resets the counted value to the starting point
- Positive Output—contains the value 1 (on) when the counting limit has been reached
- Negative Output—contains the value 0 (off) when the counting limit has been reached
- Current Value—indicates the current value of the count

Digital and Analog Values

- Preset Value—analog value that specifies the starting value
- Increment Value—analog value that specifies the amount by which the count is to increase or decrease each time
- Terminal Value—analog value that specifies the counting limit
- AutoClear—digital value that resets the count to the starting point whenever the terminal value is reached

Counting begins when another FactoryLink task, such as Math & Logic or EDI, writes a 1 (ON) to the Up Clock element. This triggers the Programmable Counters task to move the Current Value toward the Terminal Value by the Increment Value. If the Preset Value is less than the Terminal Value, the Increment is added to the Current Value. If the Preset Value is more than the Terminal Value, the Increment is subtracted from the Current Value.

Example One

In this example, counting is triggered to count bottles (20 per case). The Preset Value (start count) is 0 and the Terminal Value (count limit) for the number of bottles per case is 20. The Increment Value of 1 represents one bottle. When counting is triggered, each bottle counted increases the current count of bottles (starting with 0 in the case) by 1 until the case contains 20 bottles (until the Current Value reaches the Terminal Value of 20).

When the case contains 20 bottles (when the Current Value reaches the Terminal Value), the Counter task indicates the case is full by force-writing a 1 (ON) to the Positive Output element and force-writing a 0 (OFF) to the Negative Output element. At this point, if AutoClear = YES, the Current Value element is reset to 0 (the Preset Value) and the count can begin again. If AutoClear = NO, the current Value element remains at 20 (the Terminal Value) until another task writes a 1 (ON) to the clear element, indicating the count can begin again. The count does not continue past 20 (the Terminal Value). Each time the bottle count reaches 20 (the Terminal Value), the Counter task again force-writes a 1 (ON) and a 0 (OFF) to the Positive and Negative Output elements. When AutoClear = YES or when the Clear element is triggered, the bottle count is reset to 0 (the Preset Value), ready for a repeat of the counting process.

Example Two

You can set up another task, such as EDI or Math & Logic, to react to a deviation, such as a defective bottle, during the count by adjusting the count. To adjust the count, that task writes a 1 (ON) to the Down Clock element to cause the value of the Current Value element to move toward the Preset Value by the Increment Value.

For example, during counting, if a defective bottle is counted but not packed in the case, the EDI or Math & Logic task subtracts that bottle from the total count by writing a 1 (ON) to the Down Clock element to cause the Current Value to move toward the Preset Value (0 in this example) by the Increment Value (1 in this example).

After six bottles have been counted and packed in the case, the Counter task counts the seventh bottle. But the seventh bottle is defective, so it is not packed in the case. Therefore, the EDI or Math & Logic task subtracts that bottle from the total count by writing a 1 (ON) to the Down Clock element. This causes the Current Value to move from 7 down to 6.

- **PROGRAMMABLE COUNTERS TASK DEFINITION**

- *Principles of Operation*

-
-

If all counted bottles are defective and thus are not packed, the EDI or Math & Logic task subtracts them from the total count by causing the Current Value to count down until it matches the Preset Value (0). Although the bottle count is now 0, the Output elements have not been affected and the current counting operation continues until the case contains 20 bottles.

Report Generator Task Definition

Data FactoryLink collects or computes is stored as data elements in a real-time database. Each time data is collected or computed, the value stored in the real-time database for an element is overwritten by the new data.

If you want to report on the data, you can write the data to a report file as it is received using Report Generator. Report Generator is a flexible reporting tool that lets you define custom reports. The data included on the report can be generated as a disk file, a printed report, or exchanged with other programs that accept ASCII files.

Some typical uses for generating report data include the following:

- Predicting potential problems based on data patterns
- Reporting on productivity of shifts
- Generating hardcopy reports for management or specific agencies

This chapter introduces the operational concepts to configure Report Generator operations. Refer to “Defining the Report Format” on page 553 in the *FactoryLink Configuration Guide* for information on procedures to configure this task. Use “Report Generator” on page 467 in the *FactoryLink Reference Guide* for a quick description of this task’s panels, its entries, and its associated error messages.

- **REPORT GENERATOR TASK DEFINITION**

- *Reporting Methodology*

-
-

REPORTING METHODOLOGY

This section describes and illustrates how memory-resident real-time data is logged to a report file for generating a report.

This task completes the following steps to generate a report:

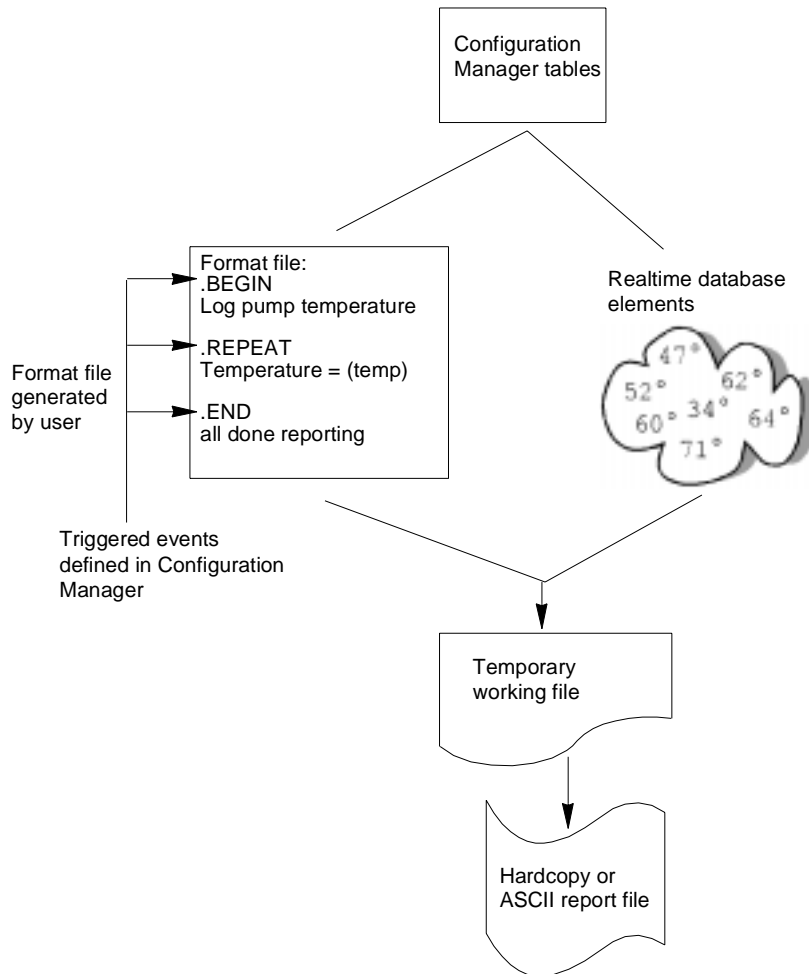
1. The real-time database receives and stores data from various sources, such as a remote device, user input, or computation results from a FactoryLink task. When data is collected and stored in this database, other tasks can access and manipulate it.
2. When a report is triggered, Report Generator reads the values of real-time database elements included on the report and maps them to object names. Object names are used in defining the report format or template file.
3. Report Generator checks the report format file to determine placement of text and objects in the report file. The format file contains keywords that trigger when the report starts, ends, and writes data. Each keyword represents a section. When the trigger executes, the associated section of the format file is processed and written to a temporary working file.
4. Report Generator uses the information in the report format file to create a temporary disk-based working file. This working file is a temporary file that remains open until the report completes and exists only until the report is completed.

The temporary file resides on disk, not in memory, to protect against loss of data. For example, if FactoryLink shuts down before Report Generator has created the report archive file, the report temporary file still exists on disk.

5. When the report is completed, the information in the temporary disk-based working file is sent to either a permanent file on disk, a printer, or a communication port.

The following illustration outlines the process of generating a report.

Figure 100-1



- **REPORT GENERATOR TASK DEFINITION**
- *Components of a Format File*
-
-

COMPONENTS OF A FORMAT FILE

User-defined ASCII format files control the look and contents of the report. A unique format file is defined for every report generated by Report Generator. This section describes the components of the format file.

Keywords

Keywords are used in the format file to trigger an action. The associated section of the format file is processed and written to a temporary working file when the trigger executes. Three keywords are used in format files:

- .BEGIN
- .REPEAT
- .END

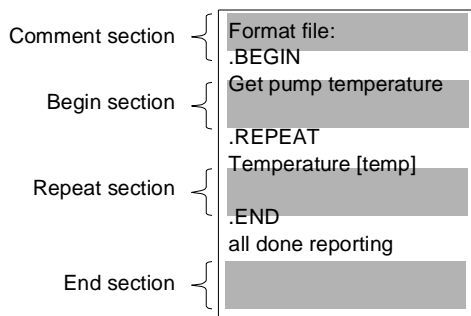
Keyword lines begin with a period (.), followed by a keyword and a line terminator such as LF (Line Feed) or CR, LF (Carriage Return, Line Feed) sequence.

Comment lines can also be included in the format file by starting the line with a period and following it with any text that does not represent a keyword. Text displayed in a comment line is not included on the report.

Sections of Format File

There are four distinct sections to the format file as shown in the following illustration.

Figure 100-2



The sections and their associated keywords are:

- **Comment section**—First section in any format file. Any text displayed in this section is not included on the report. This section usually includes a description of the report purpose. Do not precede the text in this section with a period.
- **Begin section**—Header section is delineated by the keyword `.BEGIN`, which defines the report header. Any text displayed after the `.BEGIN` keyword and before the next keyword is included at the beginning of the report.
- **Repeat section**—Repeat section is delineated by the keyword `.REPEAT`. This section contains the object names of real-time database elements to include on the report. Each time the repeat section executes, element values are written to the temporary working file. This section can also include literal text included in the report.
- **End section**—End section is delineated by the keyword `.END`. This section ends the report and contains text to include at the end of the report.

Each format file is comprised of one or more of these sections. The only required section is the end section; that is, every report format must have an end section. If you only include an end section in the format file, you generate a snapshot report. Refer to “Report Format Variations” on page 176 for more information.

- **REPORT GENERATOR TASK DEFINITION**

- *Placement of Reported Data*

-
-

PLACEMENT OF REPORTED DATA

Data specified in the format file is collected from the real-time database and placed into the report. Placement of real-time database values is determined by the following:

- Location of its object name in the format file
- Format specifiers

Location of Object Name

Object names act as a placeholder for data and are linked to data elements in the real-time database. The value of the real-time database element it is linked to replaces the object name during report generation.

Braces {} or brackets [] identify object names within the format file.

- Enclose the object name in braces {} for data that may vary in length. This places the data relative to other text in that line because the position may change based on the length of the element value. A typical use may be to locate data within a sentence.
- Enclose the object name in brackets [] for fixed position data. The value of the element associated with the object name is displayed in the report exactly where the object name is displayed. The starting bracket is the anchor for the data. This is typically used for formatting data in columns.

The identifier (braces or brackets) is not displayed in the generated report file. Use an escape sequence identified in “Escape Sequences” on page 178 if you want a brace or bracket to be displayed in the report.

You can use object names in the begin, repeat, and end sections.

Format Specifiers

Format specifiers allow you to define a variable where a literal is expected. Format specifiers can consist of two types of objects:

- Ordinary characters, which are copied literally to the output stream
- Variable specifiers, which indicate the format variable information are displayed in

Format specifiers use the following form:

% [flags][width][.prec]type

Refer to *FactoryLink Fundamentals Guide* for more information on this syntax. The following table provides a list of the specifiers typically used with Report Generator.

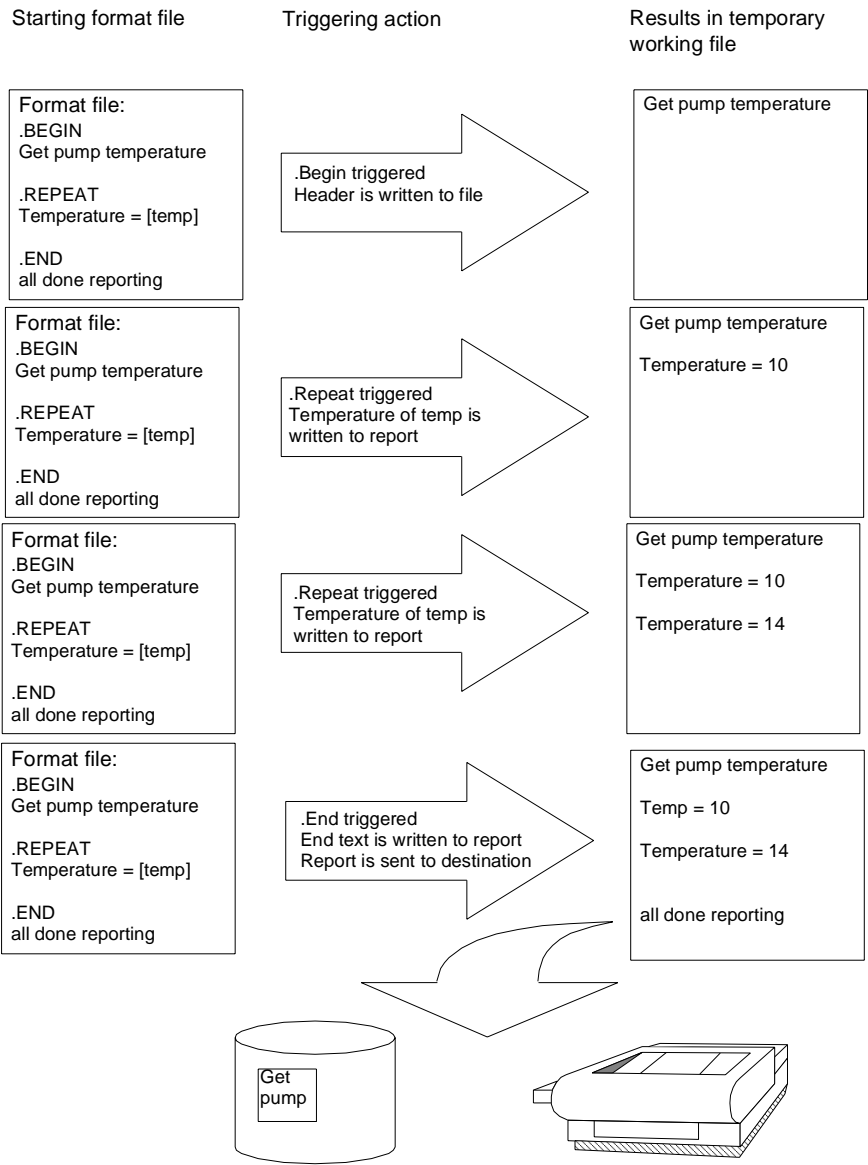
| Sample Archive File | Report Archive File Tag | Value of Archive File Tag | Actual Archive File Path Generated |
|------------------------|-------------------------|---------------------------|------------------------------------|
| C:\USCO\REPORT.%03d | A_DOY | 33 | C:\USCO\REPORT.033 |
| C:\USCO\%s.RPT | shift | "FIRST" | C:\USCO\FIRST.RPT |
| C:\USCO\PUMP%d\RPT.RPT | pump_no | 1 | C:\USCO\PUMP1\RPT.RPT |

TRIGGER ACTIONS

When a .BEGIN, .REPEAT or .END trigger executes, the associated section of the format file is processed and written to a temporary working file. The following figure illustrates what occurs when each keyword is triggered.

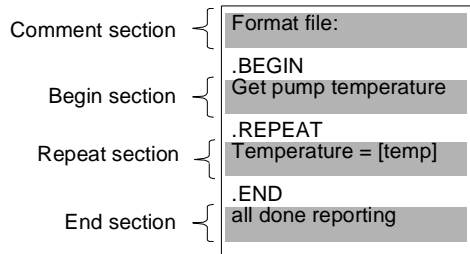
- **REPORT GENERATOR TASK DEFINITION**
- *Trigger Actions*
-
-

Figure 100-3



This example generates a report using the following format file, which includes all three sections.

Figure 100-4



This sample report format is used to generate an historical data log. A temporary working file is opened when the report is triggered. This file remains open until the end section is triggered.

The report header is written to the file when the begin section is triggered. In this example, Get pump temperature is written at the top of the report.

When the repeat section is triggered, the values of the data elements mapped to the object names included in this section are read from the real-time database and written to the file. In this example, the value of the real-time element containing the pump temperature is mapped to the object name temp and is written to the report.

Any literal text included in this section is also written to the file. In this example, the literal text Temperature = is written to the report in front of the element value.

The event that triggers the repeat can be a periodic sampling, a specific time, or an event driven trigger like a part meeting a photo-eye in a conveyor system.

You can trigger the repeat section any number of times before ending the report. In this example, the pump temperature is written to the report twice. The first time its value is 10; the next time its value is 14.

The literal text in this section is written to the temporary working file when the end section is triggered; then, the entire report is sent to its configured destination. This can be a disk-based file, a printer, or across the network to another node. The temporary working file is deleted.

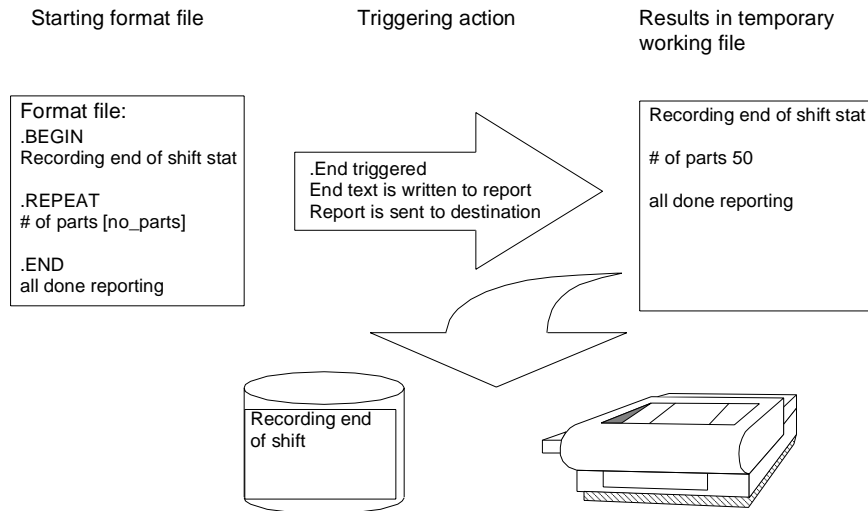
- **REPORT GENERATOR TASK DEFINITION**
- *Report Format Variations*
-
-

REPORT FORMAT VARIATIONS

A simple variation on the report used in the previous example can be generated by not specifying an event or trigger for the begin section. This is typically used if no header type information is necessary. An end section must always be specified.

Another common format for reports is a snapshot report. The purpose of this type of report is to gather information and to generate a printed report by triggering a single event. This is done by specifying only an end trigger. The begin and repeat keywords in the format file are ignored because no triggering event is specified. The end event causes all information in the format file to be sent instantly to the printer. The following figure illustrates this type of report.

Figure 100-5



COMPLETE TRIGGERS

FactoryLink returns a complete status for each of the triggered events once the operation has completed. The operation is considered completed for begin and repeat sections when the information is written into the temporary file. The operation is considered completed for the end section when the temporary file is sent to the specified device (printer, communication port, or disk file).

The complete status allows you to effectively coordinate report generator operations with other FactoryLink tasks and to display the status to the operator.

Typically the data reported on in the repeat section is generated from an external device (EDI task). In order to maintain data integrity, it may be necessary to coordinate operations between these two tasks because you do not want to log data to the report unless you are certain the data has been returned successfully from the EDI task. Likewise, you do not want to sample more data from the external device before the previous data is logged through the report generator.

Another application that may require coordination is if you want to read data from a relational database using Browser and write it to the printer. You can do this by using the complete trigger on the Browser to trigger the Report Generator repeat trigger and then have the Report Generator complete trigger generate a move to the next database row in the Browser task. This coordination takes place until all rows are fetched. Use Math & Logic to verify not only complete, but successful completion, with both tasks.

- **REPORT GENERATOR TASK DEFINITION**
- *Escape Sequences*
-
-

ESCAPE SEQUENCES

Escape sequences send instructions to the printer, such as backspaces, form and line feeds, carriage returns, or horizontal tabs. These sequences can also be used to change operating modes of printers to compressed versus standard print.

The following table lists and explains commonly used escape sequences.

| Escape Sequence | Description |
|-----------------|---|
| \b | Send backspace (0x08). |
| \f | Send form feed (0x0C). |
| \n | Send line feed (0x0A). |
| \r | Send carriage return (0x0A). |
| \t | Send horizontal tab (0x09). |
| \XX | Send 0xXX or any two uppercase hex digits (\9F). |
| \Z | Send Z, where Z is any character not previously listed. |
| \. | Send . (necessary to start a Report File line with a period.) |
| \[| Send [. |
| \{ | Send {. |
| \\ | Send a single \. |

Scaling and Deadbanding Task Definition

The Scaling and Deadbanding task (SCALE.EXE) is used to convert or scale incoming raw data to a different value range and to indicate a dead or non-recalculating band around a scaled value.

The linear scaling feature of the task is used to convert or scale the incoming raw data to a different value range. Many values read from a programmable logic controller (PLC) are in units other than those the user wishes to display, manipulate and/or archive. Use of the scaling task eliminates the need to process data through an intermediate routing mechanism and the need to write code to perform the scaling function when the scaling is linear. The scaling task, if given ranges for the incoming and desired data values, can derive the necessary conversion factor and/or offset and perform the linear scaling calculations automatically using the formula:

$$\mathbf{mx + b = y}$$

where **x** is the raw value, **m** is the multiplier, **b** is a constant and **y** is the result.

The Deadbanding task is used to indicate a band or area around a value small enough to be considered insignificant. In this case, the new value is stored and a new deadband recalculated, but the new value is not written to the program database. Since FactoryLink tasks process values upon every change, deadbanding provides a means of saving processing time and improving system efficiency.

Note: The deadbanding portion of the function cannot be implemented without configuring the scaling portion of the function.

Refer to the *Application Editor Guide* for more information on how the Scaling and Deadbanding feature works in the Application Editor.

This chapter introduces the operational concepts to configure Scaling and Deadbanding operations. Refer to “Scaling and Deadbanding” on page 219 in the *FactoryLink Configuration Guide* for information on procedures to configure this task. Use “Scaling and Deadbanding” on page 483 in the *FactoryLink Reference Guide* for a quick description of this task’s panels, its entries, and its associated error messages.

- **SCALING AND DEADBANDING TASK DEFINITION**

- *Principles of Operation*

-
-

PRINCIPLES OF OPERATION

Scaling is only available in the SHARED domain. The scaling function only applies for tags with an analog, longana or float data type.

Scaling is configured using a pair of ranges:

- one for raw values
- one for scaled values

These ranges can be specified as constants or tags. The scaling formula is adjusted accordingly if one or more of the range tags changes.

Note: If Scaling and Deadbanding is configured from the Application Editor, the system automatically assigns default tag names for raw and scaled values deadbanding and scaling lock functions, even if entered as a constant. However, if Scaling and Deadbanding is configured from within the Configuration Manager, the designer creates only specified tags.

When a value is written to a raw value tag, its related scaled value tag is updated accordingly. This is a raw-to-scaled conversion.

When a value is written to a scaled value tag, its raw value tag is updated accordingly. This is a scaled-to-raw conversion.

Prior to changing a range tag, raw value tag or scaled value tag, the function should be disabled using the Scaling Lock Tag. When the Scaling Lock Tag has a non-zero value, changes made to the tag are not propagated to their related members. After the changes to that function are made and the function is re-enabled, the current raw value is scaled and written to the scaled value tag. Any changes to the ranges are applied to the scaled value as well.

Deadbanding, which applies to raw-to-scaled conversion but not to scaled-to-raw conversion, may be specified in one of two ways:

- as an absolute (ABS) number of Engineering Units (EUs)
- as a percentage (PCT) of the scaled range

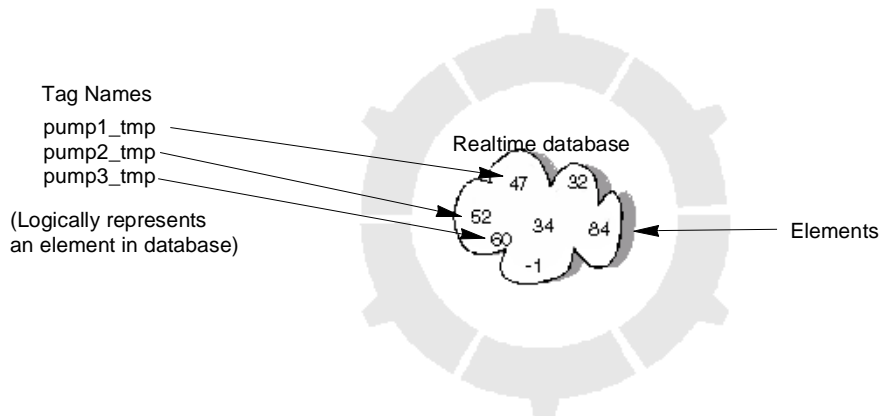
During raw-to-scaled conversion, a newly calculated scaled value that does not exceed the deadband is not written to the database. If deadbanding is being applied to a tag associated with scaling rather than a specific alpha-numeric range, deadbanding is specified by a percentage of a range rather than as an absolute value. If the deadband variance for a scaled tag is specified as an absolute value, then no deadbanding is applied to the associated raw tag.

For this example, assume the temperature of the liquid in a tank is being monitored. The tank temperature probe records raw (incoming) data on a Celsius scale. The operators monitoring the temperature are more familiar with the Fahrenheit scale and wish to have the data displayed on that scale.

The temperature never falls below freezing or above the boiling point.

DEFINING TAGS

Data stored in the real-time database is an element. Each element is assigned a logical name called a tag. This tag is used to reference the element in the real-time database.



Once an element is defined, you can make unlimited references to this element. Any FactoryLink task containing a reference to an element can read and write data to and from the element at run time.

During development, FactoryLink stores tag names in the FLAPP directory in the object database table. This information is updated to the .CT files when the run-time application is started.

Some tags are already defined in FactoryLink when it is shipped. Others are defined during application development either within the Configuration Manager or the Application Editor.

This chapter describes how to define tag names for database elements and provides some suggestions on how to use tags in your application.

- **SCALING AND DEADBANDING TASK DEFINITION**

- *Tag Naming Guidelines*

-
-

TAG NAMING GUIDELINES

The following guidelines apply when you assign a tag name:

- 1-32 characters long

If using an array, add delimiters of up to 16 characters.

Refer to “Defining Element Arrays” on page 207 for more details.

If using PowerNet, use the model:

32 characters less 7 characters = 25 characters.

If using Scaling and Deadbanding, use the model.

32 characters less 7 characters less 9 characters.

Refer to the *Application Editor Guide* for details on the added extensions.

- Valid characters are A-Z, 0-9, _, @, \$, ., :
- Do not start with a number
- No embedded spaces

System Added Extensions

When defining tag attributes in the Tag Definition dialog, new tags are created automatically from the originally defined tag name if you are using PowerNet or Scaling and Deadbanding.

PowerNet

When a new tag is created, an extension of up to 6 characters plus a dot (7 characters maximum) is added. These additional characters reduce the maximum length of the original tag name; therefore, the maximum effective length of the original tag name is 32 less 7, or 25. If the original tag name plus extension exceeds 32 characters, a warning is issued.

Scaling and Deadbanding

When a new tag is created, an extension of up to 8 characters plus a colon (9 characters maximum) is added. These additional characters reduce the maximum length of the original tag name; therefore, the maximum effective length of the original tag name is 32 less 7 less 9, or 16.

If you are not sure if you will be using PowerNet or Scaling and Deadbanding, you may choose to define tag names using only 16 characters. Also, remember if shortening tag names, do not reduce the length of the extensions.

Schema Task Definition

FactoryLink relational databases are configured in a table format consisting of rows and columns. The schema of the table defines the number, size, and content of the rows and columns.

This chapter introduces the operational concepts to configure Schema operations. Refer to “Defining Schemas” on page 225 in the *FactoryLink Configuration Guide* for information on procedures to configure this task. Use “Schemas” on page 493 in the *FactoryLink Reference Guide* for a quick description of this task’s panels, its entries, and its associated error messages.

Database Logging

Data is logged to relational databases in a table format that allows data collection and organization by category. Examples of data categories are factory location and data collection times. The following figure illustrates relational database organization using the category employee start date.

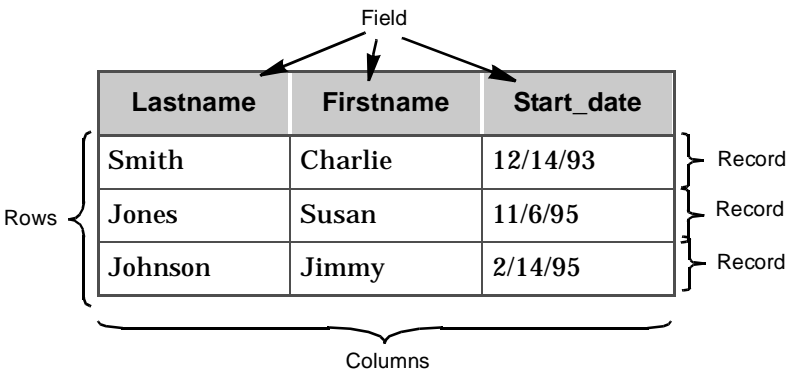


Figure 102-1 Relational Database Organization

- **SCHEMA TASK DEFINITION**
-
-
-

The data in a single row is related. Each entry in the row forms a single record. The example table contains three records.

The data in a table column represents a record field. For example, the example table has three fields:

- One contains employee last name (Lastname)
- One contains employee first name (Firstname)
- One contains employee start date (Start_date)

The fields in a database table and their attributes, such as data types and field lengths, define its structure.

Relational databases contain one or more tables. Each table in dBASE IV equates to a file, and each table or file has a unique name.

Each unique table structure used to log data is assigned a name in the Schema Control panel for Database Logging. The columns and attributes in each table are specified in the Schema Information panel for each schema.

Create a table index if you want to retrieve the records in an order other than the order they were added to the table. Specify the desired column name(s) in the Index Information panel to create an index.

The Security Logging Schema panel is used to define table structure for the Security Log if security has been configured for any events.

Data-Point Logging

The schemas for four tables for Data-Point Logging are predefined.

| Schema Name | Data Type |
|--------------------|------------------|
| TAGDATA | float. |
| SMALLINT | smallint. |
| LARGEINT | integer. |
| FLOATVAL | float. |

The table size, column structure, and data types used for any Data-Point Logging table are adjusted using the Data-Point Schema Control panel. Add the name of the new table to the Data-Point Logging Control panel when you create an additional Data-Point Logging table.

Database vs. Data-Point Logging

FactoryLink depends on the data logging method used for defining a table structure schema. Prior to configuring the logging function you should determine which logging task you will be using for a particular situation.

Database Logging is best for situations when you want to

- Log all tags when any tag changes or is triggered
- Simultaneously log a group of like (logically associated) tags
- Group data logged based on some criteria
- Configure a table column to be a dynamic pen on a trend chart

Data-Point Logging is best for situations when you want to

- Log a tag only when its value changes
- Use preconfigured tables and eliminate the time spent setting up tables
- Be able to index on log time or tag name or both
- Sort all logs of a tag in order of occurrence
- Configure a tag to be a dynamic pen on a trend chart
- Dynamically change the list of tags being logged during run time

For example, if you have a tank you are monitoring the process temperature in with a single probe in the tank, you would use Data-Point Logging to track the tag value. If you have six probes in the same tank and you want to see the value returned from each probe at a given point in time, you would use the Database Logging task.

Refer to FactoryLink ECS Fundamentals Guide Chapter 87, “Database Logging Task Definition” and Chapter 88, “Data-Point Logging Task Definition” to determine which logging method to use for your application before configuring a table.

- **SCHEMA TASK DEFINITION**

-
-
-

Trending

The referenced database table must be configured in the Trend Database Tables panel and either the Database Logging function or the Data-Point Logging function if you are using the Trend database. The Data-Point and Database Trend tables are mutually exclusive; therefore, a single table cannot be configured in both functions.

Database Logging Schemas

When you log data to FactoryLink Database Logging, in addition to logging the data, you may find it useful to:

- Group certain data together (grouping)
- See the order the data was logged or the actual time a data event occurred (sequencing or ordering)
- Sort the records into a particular order, such as alphabetical, before viewing them (indexing)

When planning your database tables, determine if you will use these options so you can specify appropriate columns.

Grouping Data

When you log data with a group association, you can log more than one set of data to the same table if you create a column that contains the group associated with each row of data.

Ordering Data

FactoryLink permits you to log grouped and non-grouped data with or without a sequence or order number. If you elect to associate a sequence number with rows of data, your table structure must include a column that contains the sequence number associated with the row of data.

Indexing

Records are displayed in a relational database table in the order they were added. In some cases, you may want to create an index to retrieve the records in another order. An index uses the values in one or more of the fields to reorganize the records in a database file. The field or fields selected for indexing are called the index key(s).

When a table is indexed, an index file containing the indexed order is created; however, the actual record order in the database file remains unchanged. Accessing indexed records is faster than accessing non-indexed records.

Assigning Names to Schemas

Schema names are assigned on the Schema Control panel and each schema definition can be used as the structure for more than one table. Define only as many unique structures as you use because schema definitions are shared.

For example, assume you log the last name, first name, and start date of your employees to two different tables. One table contains information about employees in the Dallas facility and the other contains information about employees in the Atlanta facility. Only one schema is defined because both table structures are the same.

Dallas table

| Lastname | Firstname | Start_date |
|----------|-----------|------------|
| Smith | Charlie | 12/14/93 |
| Johnson | Jimmy | 2/14/95 |

Atlanta table

| Lastname | Firstname | Start_date |
|----------|-----------|------------|
| Jones | Susan | 11/6/95 |

Table 102-2

Same schema

- **SCHEMA TASK DEFINITION**
-
-
-

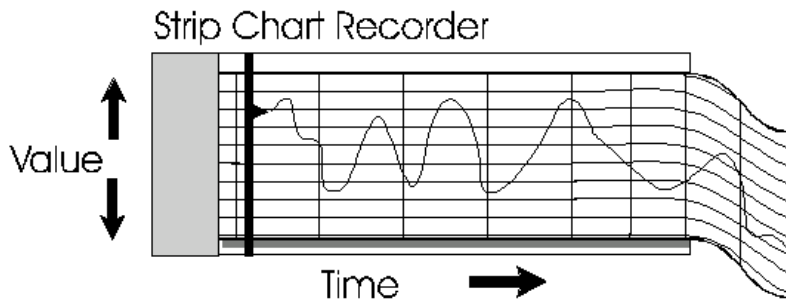
Trending Task Definition

As data is collected or computed by FactoryLink, it is stored as a data element in a real-time database. Each time data is collected, or computed, the new data overwrites the value stored in the real-time database for an element.

Using the Trend application, you can create animated graphs called trend charts that graphically show this numeric data. To show the data as it occurs, you create real-time trend charts using the real-time database. To show the data after it occurred, you create historical trend charts using a relational database.

Trend charts resemble strip charts. The illustration below shows a typical strip-chart recorder:

Figure 103-1



This chapter introduces the operational concepts to configure Trending operations. Refer to “Configuring Trend Panels” on page 409 in the *FactoryLink Configuration Guide* for information on procedures to configure this task. Use “Trending” on page 511 in the *FactoryLink Reference Guide* for a quick description of this task’s panels, its entries, and its associated error messages.

- **TRENDING TASK DEFINITION**

- *Trending Methodology*

-
-

TRENDING METHODOLOGY

Trending is logically divided into two categories:

- real-time
- historical

These categories determine the database from which the trending data is read.

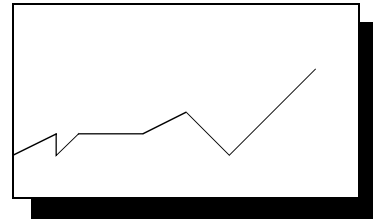
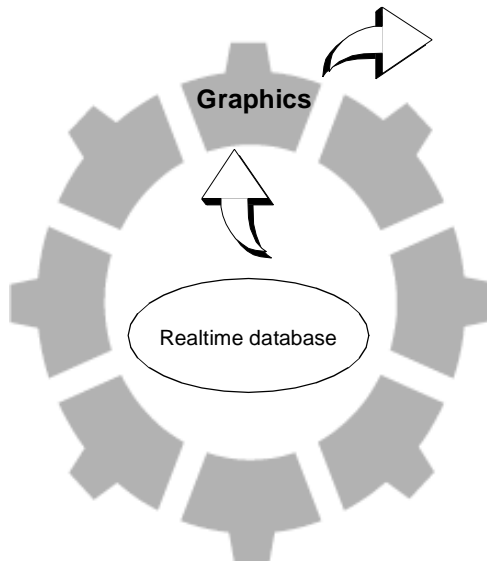
Real-time Only Trending

Real-time only trending allows you to chart data directly from the real-time database. As a new value is written to the database, the value is displayed on the real-time trend chart. Data displayed on a real-time trend chart is not saved.

Real-time only trending is a function of the run-time graphics application; therefore, refer to the *Application Editor Guide* for additional information. Following is a sequential list and illustration of events explaining the real-time trending operation:

1. The real-time database receives and stores data from various sources, such as a remote device, user input, or computation results from a FactoryLink task. When data is collected and stored in this database, other tasks can access and manipulate it. When new data is received the older data is overwritten.
2. Graphics displays the trend chart.

Figure 103-2



The Graphics task receives data and plots it to a trend chart. No other tasks are involved.

Historical Trending

Historical trending allows you to chart data from the relational database historically or in real time:

Following is a sequential list and illustration of events explaining the historical trending operation.

1. The real-time database receives and stores data from various sources, such as a remote device, user input, or computation results from a FactoryLink task. When data is collected and stored in this database, other tasks can access and manipulate it. The older data is overwritten as new data is received.
2. Logger reads the value of specific data elements stored in the real-time database and maps the data elements to columns in a disk-based relational database table.
3. Using an SQL INSERT statement, Logger sends the data from the real-time database to the disk-based relational database, via a Historian mailbox.
4. Historian inserts the data in the relational database file. Once in this file, the data can be used by other applications.

- **TRENDING TASK DEFINITION**

- *Trending Methodology*

-
-

5. Trend requests the data from the relational database, via Historian, so it can be displayed on a trend chart.
6. Trend sends the data to the run-time Graphics task.
7. Graphics displays the trend chart.

Figure 103-3

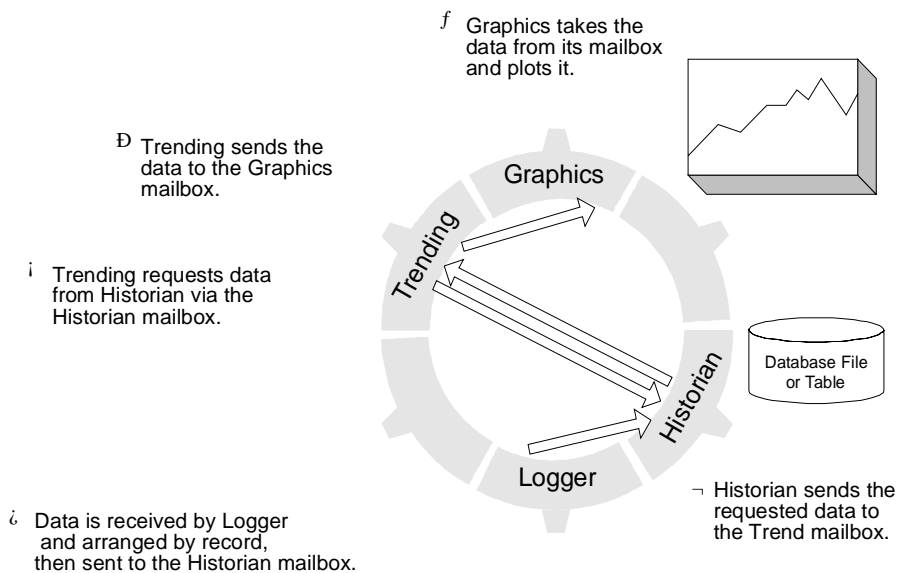


CHART TYPES

Within trending there are two types of charts:

- time-based
- event

The chart type to use depends on how the data was indexed. This can be one of the following:

- Time-stamping—If the data is indexed on a time-stamping, you must trend the data on a time-based chart.
- Integer—If the data is indexed on an integer, you must trend the data on an event chart.
- Integer plus a group ID—If the data is indexed on an integer plus a group identifier, you must trend the data on an event chart.

Refer to *Logging* in the *FactoryLink Configuration Guide* for additional information on indexing.

Time-based Chart

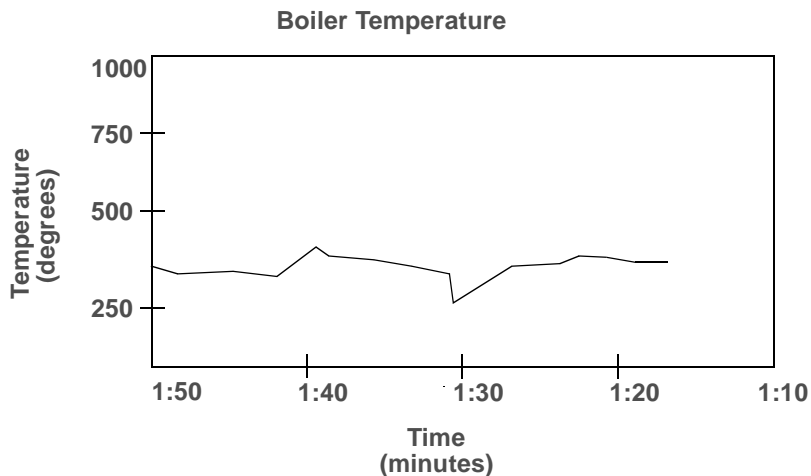
Data indexed on a time-stamping can be shown on a time-based trend chart. Time-based charts are best suited for continuous types of data. For example, using a time-based chart, you can show trends or variances in a boiler temperature over time. Because the boiler must maintain a particular temperature, the temperature is sampled every minute. Refer to the following example for a representative time-based trend chart.

- **TRENDING TASK DEFINITION**

- *Chart Types*

-
-

Figure 103-4



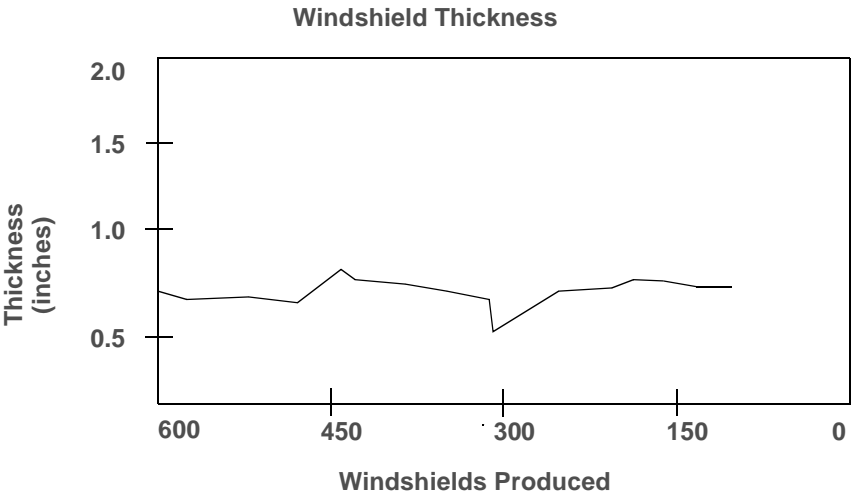
Event Chart

Data indexed on an integer can be shown on an event trend chart. The integer index indicates the order the data was logged in. Event charts are well-suited for two types of data:

- per piece (sample)
- batch (group)

Per piece data is data collected for every item in a process. For example, a manufacturer of car windshields inspects the thickness of every completed windshield as it comes off the assembly line. Using an event chart, this manufacturer can graphically represent the thicknesses of the windshields produced. Refer to the following example for a representative event trend chart.

Figure 103-5

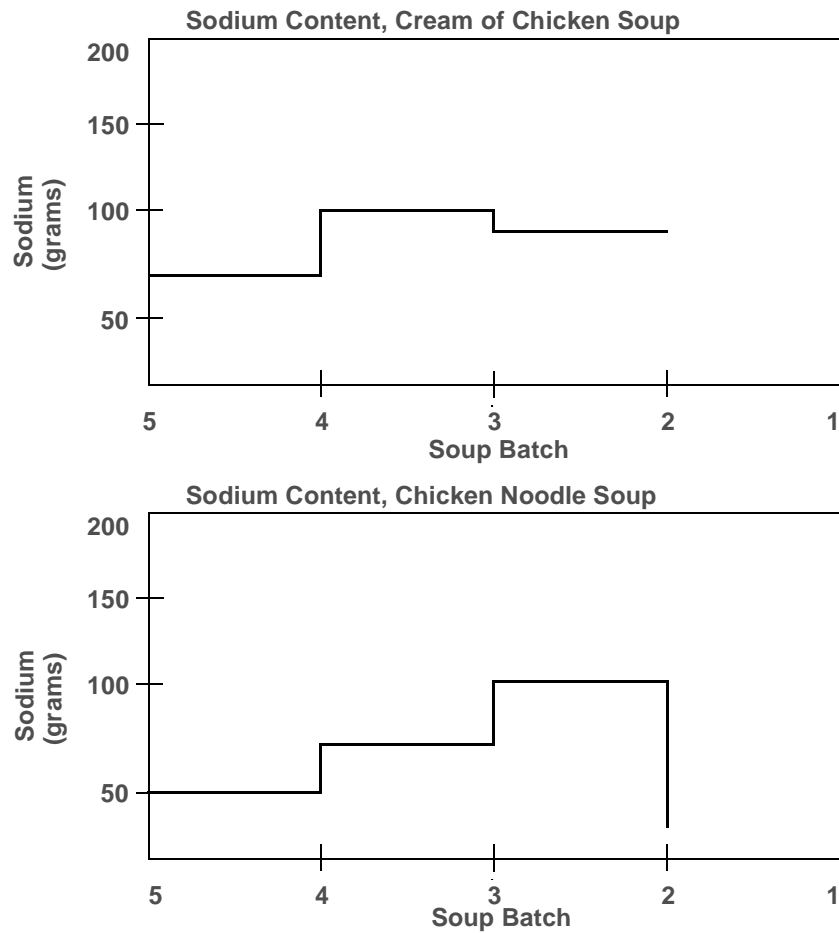


Group data is data that logically belongs together. For example, a soup manufacturer that makes two flavors of soup may want to track different batches of both flavors. Using an event chart with groups (soup flavors), this manufacturer can graphically represent the differences in sodium content for each group by batch. At the end of the batch cycle, a trigger initiates the sampling of the sodium content for the batch. This sample is written to the database and the sequence number increments to prepare for the sampling of the next batch. The differences can be shown as separate trend charts for each soup flavor: cream of chicken or chicken noodle.

Refer to the following example for representative trend charts.

- **TRENDING TASK DEFINITION**
- *Chart Types*
-
-

Figure 103-6



SWITCHING BETWEEN REAL-TIME AND HISTORICAL MODE

Switching modes allows you to switch between viewing real-time and historical data. When in real-time mode, a trend chart shows data as it is being logged to the database.

When in historical mode, a trend chart shows only data stored in the database. If you change screens while in historical mode and then return to the trending screen, FactoryLink reloads the trend chart with the same historical data you were viewing before you changed screens.

ZOOMING AND PANNING

Because historical trend data is written to a relational database, it can be arranged to show different data ranges (pan) and these ranges expanded or collapsed (zoom). Using FactoryLink, you can zoom and pan to a particular sample or group in event charts, or time in time-based charts.

Panning allows you to select the time span of data to display in a historical trend chart. Using this feature, you can move forward or backward through historical data and you can move to a specific time or sample.

Zooming either doubles the amount of data displayed or decreases the amount displayed by half. You see the same number of points, but you zoom in for a wider start/end range or zoom out for a narrower range.

Pan and zoom fields animated as input-text fields should not be set for background updates. If they are and you disable and enable windows with trend charts in historical mode, the pan and zoom settings are executed. This does not occur when changing the drawing within a window.

PEN TYPES

Using FactoryLink, you can create either fixed or variable pens. Fixed pens allow you to permanently assign a database column to a particular pen. Variable pens allow you to assign the column to a pen at run time. You can assign multiple pens to a trend chart.

Note: We recommend you configure no more than eight pens to a chart for good readability.

- **TRENDING TASK DEFINITION**

- *Value Cursor*

-
-

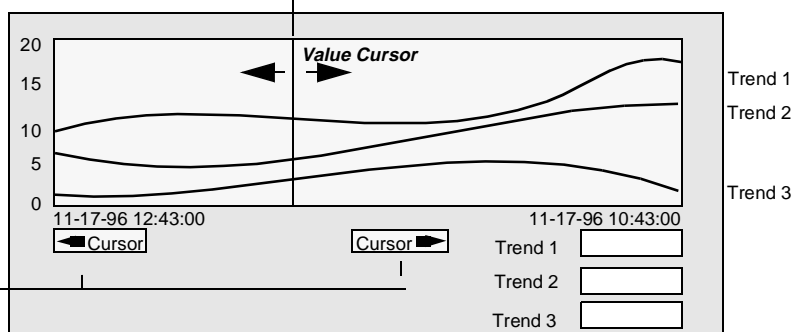
VALUE CURSOR

A value cursor allows you to display the value associated with a point on a trend chart. When you click anywhere in the chart at run time, the value cursor, which looks like a vertical bar, is displayed. The system displays the values for the points where the value cursor intersects the trend lines. Refer to the following example for a representative value cursor.

Figure 103-7

The value cursor indicates the selected point as it intersects the trend lines.

Move the value cursor back and forth to see the value at different points along the trend lines.



The values of the trend lines at the cursor are displayed in output-text fields.

You can determine the time an event on an event trend chart was logged as well as its value using a value cursor.

CONFIGURING PROGRAM ARGUMENTS

You can configure the following system configuration program arguments to affect Trend functions:

- L or -l Enables logging of errors to the log file. By default, Trend does not log errors.
- V[1-4] or -v[1-4] Writes the trend chart events to terminal output. If logging is enabled, the trend chart events are written to the log file. A verbose level of 1 generates minimum data, and a level of 4 generates the most data. Level 4 writes all the values of all the points for all the pens in a chart. This level can generate a tremendous amount of data.
- W[5-300] or -w[5-300] Sets the maximum timeout in seconds for Trend to wait for a response from the Historian task. The default is 30 seconds.

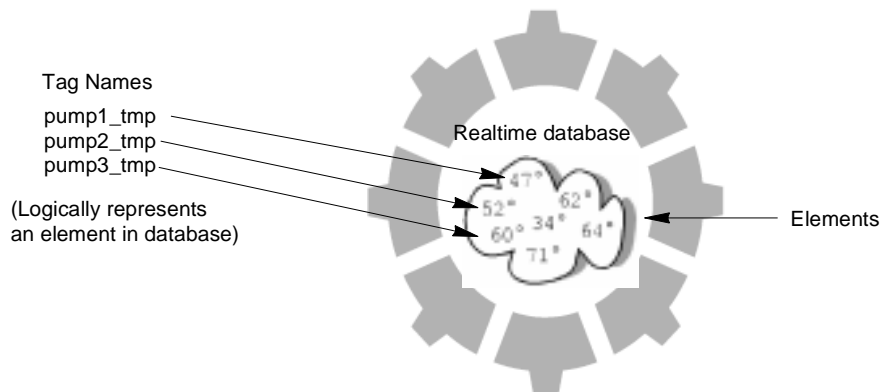
Perform the following steps to configure one or more arguments:

- 1 Ensure the current domain selected is USER on the Configuration Manager Main Menu in the Domain Selection box.
- 2 Choose the System Configuration option to display the System Configuration Information panel.
- 3 Enter one or more arguments, separated by spaces in the Program Arguments column for the Trend task.

- **TRENDING TASK DEFINITION**
- *Configuring Program Arguments*
-
-

Defining Tags

Data stored in the real-time database is an element. Each element is assigned a logical name called a tag. This tag name is used to reference the element in the real-time database.



Once an element is defined, you can make unlimited references to this element. Any FactoryLink task containing a reference to an element can read and write data to and from the element at run time.

During development, FactoryLink stores tag names in the FLAPP directory in the object database table. This information is updated to the .ct files when the run-time application is started.

Some tags are already defined in FactoryLink when it is shipped. Others are defined during application development either within the Configuration Manager or the Application Editor.

This chapter describes how to define tag names for database elements and provides some suggestions on how to use tags in your application.

- **DEFINING TAGS**
- *Tag Naming Guidelines*
-
-

TAG NAMING GUIDELINES

Valid tag names conform to the following grammar:

[<*node*>:]<*name*>[<*dims*>][.<*ext*>]

where

<*node*> Is limited to 8 characters.

<*name*> Is limited to 32 characters.

If using an array, add delimiters of up to 16 characters. Refer to "Defining Element Arrays" on page 207 for more details.

If using PowerNet in your enterprise, the **character count is reduced to 23**.

When PowerNet is used, a logical name of up to 8 characters plus a colon (9 characters maximum) is added to reference the FactoryLink application that is the source of the real-time database element. These additional characters reduce the maximum length of the original tag name by up to 9, making the maximum length of the original tag name 23.

Refer to *Communications in the FactoryLink Configuration Guide* for more details.

If using Scaling and Deadbanding, the character count is reduced to 25.

When Scaling and Deadbanding is applied to a tag, a FactoryLink-generated extension of up to 6 characters plus a dot (7 characters maximum) is added. These additional characters reduce the maximum length of the original tag name by up to 7, making the maximum length of the original tag name 25.

Refer to "Scaling/Deadbanding Definitions" in the *Application Editor Guide* for details on the added extensions.

If using Scaling and Deadbanding with PowerNet in your enterprise, the character count is reduced to 16.

If you are not sure if you will be using PowerNet or Scaling and Deadbanding, we recommend you define tag names using only 16 characters.

<*dims*> Is limited to 16 characters.

<*ext*> Is limited to 16 characters.

The maximum tag name length, including *<node>*, *<name>*, *<ext>*, and the separators *:* and *.* is 32. Legal characters for the *<node>*, *<name>*, and *<ext>* strings are

- {A-Z}
- {a-z}
- {0-9}
- {@\$_}

These strings cannot begin with a number or contain spaces.

Defining Tags While in the Configuration Manager

When using the Configuration Manager during application development, some fields require the name of a tag. If you enter the name of a tag not already defined in the application, the following Tag Definition dialog is displayed.

Name assigned
to tag →

The screenshot shows a 'Tag Definition' dialog box. The 'Tagname' field contains 'DYNLOGCOMMAND'. The 'Description' field contains 'Dynamic Logging command message tag for Data Point Logger'. There are empty fields for 'Type' and 'Domain'. Below these are fields for 'Size', 'Array Dimensions', and 'Length' (set to 512). A 'Default Value' field is also present. A 'Persistence' section contains three groups of checkboxes: 'Use Domain Settings' (checked), 'Saving' (with 'On Time' and 'On Exception' options), and 'Restoring' (with 'Set Change Status On' and 'Set Change Status Off' options). At the bottom are 'OK', 'Cancel', 'Edit', and 'Help' buttons.

If you specified multiple tags on a configuration panel, this panel will display each tag name in turn until all tags are defined.

Complete the following information for each tag.

Tag Name This field displays the name of the tag you are defining. Tag names can be up to 32 alphanumeric characters with no embedded spaces or periods. Do not start the name with a number.

- **DEFINING TAGS**
- *Tag Naming Guidelines*
-
-

| | |
|-------------|---|
| Description | Enter a description of up to 79 characters that defines the purpose of the tag. |
| Type | <p>Enter the type of data that will be stored in this tag. This can be one of the following, although you may be restricted to a subset of these depending on where the tag is specified. Refer to the <i>FactoryLink Configuration Guide</i> for valid data types for specific fields.</p> <div> <div>digital</div> <div>Digital is a binary data type. Its value can be 0 or 1.</div> </div> <div> <div>analog</div> <div>Analog is a 16-bit, signed integer. Its value can range between plus or minus 32,768.</div> </div> <div> <div>longana</div> <div>Long analog is a 32-bit, signed integer. Its value can range between plus or minus 2³¹-1.</div> </div> <div> <div>float</div> <div>Floating-point is an IEEE double precision number with 31 places to the right of the decimal.</div> </div> <div> <div>message</div> <div>A message can be any combination of alphanumeric characters. Its length is controlled by its tag definition.</div> </div> <div> <div>mailbox</div> <div>Variable length data organized as a queue.</div> </div> |

The remaining fields on the Tag Definition dialog are used for defining tag persistence. With tag persistence activated, the value of the tag is periodically saved to a disk file. If a value exists in this file for a tag, it is written to the tag when the task is started if the -r flag is set for FLRUN. In this way, you do not lose important information by exiting the task.

- | | |
|---------------------|---|
| Use Domain Settings | Choose this option if you want to use persistent settings defined for the domain. If you want to define tag-specific persistence, deselect this option (box does not contain an x). Then set the following persistent options for this tag. |
| Saving | <p>Controls how often the element's value is saved to a disk file. This can be one or both of the following. If you choose both options, the element's value is saved based on a time trigger and when it changes.</p> <p>On Time—Choose this option if you want the tag value saved based on a time trigger.</p> <p>On Exception—Choose this option if you want the tag value saved if the tag value changes.</p> |
| Restoring | <p>Controls whether the element's change status flag is set to on or off when restoring the saved value to the element. This can be one of the following.</p> <p>Set Change Status On—The element's change status flag is set to on when restoring the saved value.</p> <p>Set Change Status Off—The element's change status flag is set to off when restoring the saved value. Use this option to prevent activation of digital triggers on restart.</p> |

Defining Tags While in the Application Editor

You can also define a tag name while developing the application within the Application Editor. The same information is included in the dialog as well as many additional items configured in the various Configuration Manager tasks. Refer to the *Application Editor Guide* for details.

- **DEFINING TAGS**
- *Tag Naming Guidelines*
-
-

Editing a Tag

If a tag needs to be edited after it is created, access the Tag Definition dialog by pressing CTRL + T.

Click on Edit at the bottom of the window to edit general information for the tag.

When the edit function is selected, the attributes of the tag definition become editable. Changes to the information for the tag must be made before any other changes can be executed.

If the tag is a SHARED domain tag already used in the SHARED domain, the tag domain cannot be changed.

Array dimensions can be increased from the existing dimensions or decreased if the items being eliminated have not already been referenced in FactoryLink. If a tag was not originally created as an array, it cannot be changed to be an array tag. Likewise, the number of dimensions cannot be changed. For example, tag [3][4] cannot be changed to tag [3].

When all changes to the tag definition are complete, click on OK at the bottom of the dialog.

When all editing to information for the tag is completed, click on OK to save all the options.

DEFINING ELEMENT ARRAYS

A tag name can be assigned to a single element or a group of elements (sometimes called an array). This permits you to create multiple tags with a single operation. All the tags receive the same tag definition.

You specify an array by entering a value in the Array Dimensions field of the Tag Definition dialog. This value defines the number of elements included in the array.

Specifies the
number of elements
in the array →

The image shows a 'Tag Definition' dialog box with the 'General' tab active. The 'Tagname' field contains 'Tagname'. The 'Array Subscript' field is empty. The 'Description' field is empty. The 'Type' is set to 'ANALOG' and the 'Device' is set to 'SHARED'. The 'Array Dimensions' field is empty, and an arrow from the text 'Specifies the number of elements in the array' points to it. The 'Message Length' is set to '0'. The 'Default Value' field is empty. There is a checkbox for 'Follow Domain Persistence' which is unchecked. Buttons for 'Apply', 'Cancel', 'Help', 'OK', and 'Cancel' are visible at the bottom.

If you specify a value in the Array Dimensions field, multiple elements are created in the real-time database. Once created, each element can be referenced individually. The value you specify in the Array Dimensions field depends on whether you are defining a single-dimensional or multi-dimensional array.

Defining One-dimensional Arrays

For one-dimensional arrays, specify a single number in the Array Dimensions field. One-dimensional array names take the form

`tagname[n]`

where

- `tagname` is the name defined in the Tagname field of the Tag Definition dialog.
- `[n]` is a unique number assigned to each element in the array starting with 0. Each number is surrounded with brackets [].

- **DEFINING TAGS**
- *Defining Element Arrays*
-
-

For example, if you specify 3 in the Array Dimensions field for a tag named temp, the following three elements are created in the real-time database.

```
temp[0]
temp[1]
temp[2]
```

The array numbering starts counting at 0 –. If you specify 4, you receive 5 elements: 0, 1, 2, 3, 4. Therefore, you must specify a number less than the actual number you would like to define.

Defining Multi-dimensional Arrays

For multi-dimensional arrays, specify multiple numbers separated by a comma in the Array Dimensions field. Multi-dimensional array names take the form

```
tagname[n,1][n,2]...
```

where

- | | |
|---------|--|
| tagname | is the name defined in the Tagname field of the Tag Definition dialog. |
| [n,1] | is a unique number representing the first dimension in the array, assigned to each element starting with 0. Each number is surrounded by brackets []. |
| [n,2] | is a unique number representing the second dimension in the array assigned to each element starting with 0. Each number is surrounded by brackets []. |

For example, if you specify 3,2 in the Array Dimensions field for a tag named msg_tag, the following six elements are created in the real-time database.

```
msg_tag[0][0]
msg_tag[1][0]
msg_tag[2][0]

msg_tag[0][1]
msg_tag[1][1]
msg_tag[2][1]
```

The array numbering starts counting at 0 –. If you specify 4, you receive 5 elements: 0, 1, 2, 3, 4. Therefore, you must specify a number one less than the actual number you would like to define.

Maximum Number of Arrays

This section describes the largest number of elements that can be created for each array. This depends on the platform and on the element's data type.

For Windows NT, OS/2, and UNIX

Each array can have up to 65,535 elements.

For Windows 95

An array can have as many elements as will fit into a 65,535-byte memory segment. This number depends on the element's data type. The following table lists the maximum number of elements permitted in an array for each data type.

Table 104-1 Windows Array Maximums

| Data Type | Maximum Elements |
|----------------|-------------------------|
| Digital | 4,095 |
| Analog | 3,640 |
| Long analog | 3,276 |
| Floating-point | 2,730 |
| Message | 2,730 + storage of data |
| Mailbox | |

- **DEFINING TAGS**
- *Using Tags as Triggers*
-
-

USING TAGS AS TRIGGERS

Many FactoryLink tasks use tags to trigger actions. The examples below are just a few of the possibilities:

- You can use the complete status tag in a logger operation to initiate a different logging operation.
- You can use the same tag to trigger multiple operations. For example, you can define a single tag that triggers multiple operations at the start of each hour.
- Triggering can be based on the fact that an element's value has changed.
- Triggering can be based on a combination of the trigger element's value and the value of its change-status flag. For example, if you are using a digital element to trigger an action, its change-status flag must be 1 (ON) and its value must be 1 (ON) for the operation to be triggered.
- You can define triggers using the Interval and Event timer module.
- You can configure function keys as triggers.
- You can use the Math & Logic task to set triggers of any data type, except mailbox, to start an operation.

PREDEFINED ELEMENTS

See the current FactoryLink Release Notes for a complete listing of the FactoryLink pre-defined elements.

Using the Configuration Manager

The Configuration Manager is the user interface to the development environment. You can use the Configuration Manager to perform the following actions:

- Designate how tasks and elements are used at run time
- Specify elements to be used by each task at run time
- Review and modify elements during configuration
- Review and modify application data

The development environment is a series of configuration tables; one or more tables exist for each FactoryLink module. You enter data in these configuration tables through panels. There are two types of panels in FactoryLink.

- **Structured**—Structured panels contain rows and columns. Each row represents an entry in the table (sometimes called a record). Each column has a heading, and represents a field in a record.
- **Text-entry**—Text-entry panels have an open form in which ASCII text can be entered.

This chapter describes how to open and use the Configuration Manager. The discussion in this chapter assumes you know how to use a mouse or other pointing device to:

- Position the cursor
- Choose an item
- Choose a range of text

- **USING THE CONFIGURATION MANAGER**

- *Opening the Configuration Manager*

-
-

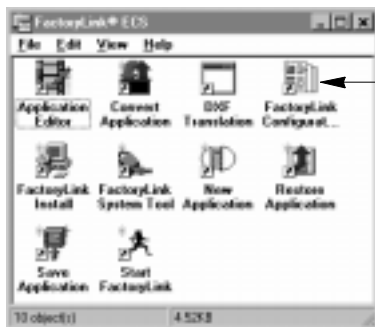
OPENING THE CONFIGURATION MANAGER

How you open the Configuration Manager depends on your operating system platform. Once opened, the Configuration Manager acts the same for all platforms.

Windows NT or Windows 95 Platform

From the Program Manager

- 1 Click the FactoryLink icon from the Program Manager window to display the FactoryLink IV program group.



If you have more than one FactoryLink application, you may have more than one FactoryLink IV program group. Be sure to open the program group for the desired application.

- 2 double click on the FactoryLink Configuration icon.

From the Command Line

- 1 Set the FactoryLink application environment variable by entering the following command at the system prompt.

Set FLAPP = **c:/flapp**

- 2 Enter the following command to open the Configuration Manager.

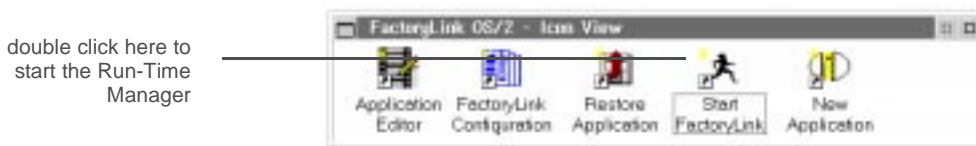
flcm

OS/2 Platform

Open the Configuration Manager in one of two ways: either from the Desktop Manager or from the command line if you are running FactoryLink on an OS/2 operating system. Both methods are described below.

From the Desktop Manager

- 1 Choose FactoryLink OS/2 from the Desktop Manager menu to display the applications available with the FactoryLink group.



- 2 double click the left button to open the Configuration Manager.

From the Command Line

- 1 Set the FactoryLink application environment variable by entering the following command from the system prompt.

Set FLAPP = **c:/flapp**

- 2 Enter the following command to open the Configuration Manager.

flcm

UNIX Platform

- 1 Be sure the FactoryLink application directory environment variable, FLAPP, is set to the full path name of the directory containing the application you are running. If it is not, set it using the following instructions.

If you are using a C shell, enter the following command at the system prompt.

setenv FLAPP *flapp_dir*

where *flapp_dir* is the full path of the directory containing the application. For example,

setenv FLAPP /usr/newapp

- **USING THE CONFIGURATION MANAGER**

- *Opening the Configuration Manager*

-
-

If you are using a Korn or Bourne shell, enter the following command set at the system prompt.

```
FLAPP=flapp_dir  
export FLAPP
```

where flapp_dir is the full path of the directory containing the application. For example,

```
FLAPP=/usr/newapp  
export FLAPP
```

2 Enter the following command at the system prompt to open the Configuration Manager.

```
flcm
```

CONFIGURATION MANAGER DISPLAY

When you open the Configuration Manager, the following screen is displayed.



This screen has three major components:

- **Menu bar**—The menu bar provides access to a group of tools you can use to manipulate or review application data.
- **Main Menu**—The Main Menu provides access to configuration tables for each FactoryLink module installed on your system. double click the name to choose a module. Refer to “FactoryLink Modules” on page 29 for a list of possible modules.
- **Domain selection**—The domain selection menu controls which domain you are opening when you choose a module from the Main Menu. This can either be shared or user. The default is the same as the domain specified in the FLDOMAIN environment variable. Refer to “Overview” on page 21 for a discussion on FactoryLink environment variables.

Double click on the desired domain or be sure to press Enter after choosing a domain, or the domain does not change. Refer to “Overview” on page 21 for details on which domain to select for each application.

- **USING THE CONFIGURATION MANAGER**

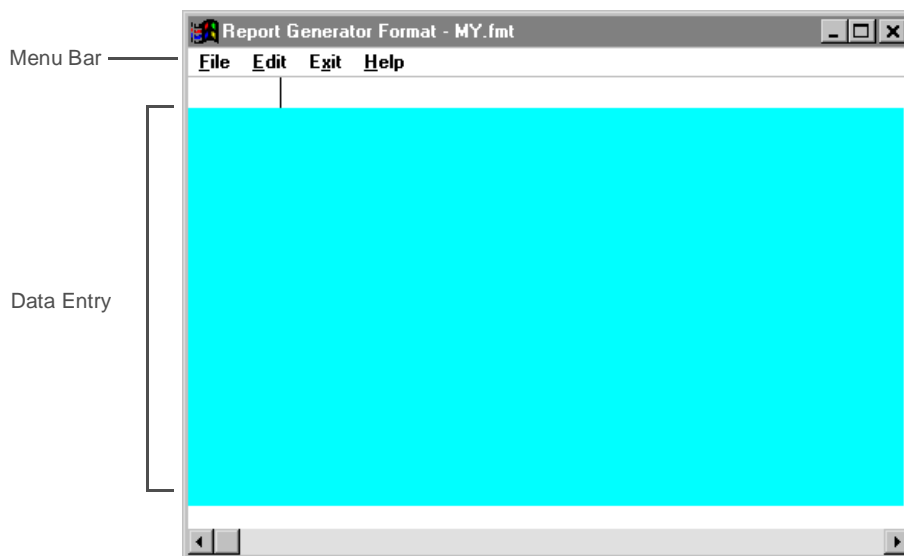
- *Working with Text-entry Panels*

-
-

WORKING WITH TEXT-ENTRY PANELS

Open configuration tables by selecting a module from the Main Menu. Data is entered in configuration tables through either structured panels or text-entry panels. This section discusses text-entry panels and their characteristics.

Text-entry panels do not have defined data-entry fields.



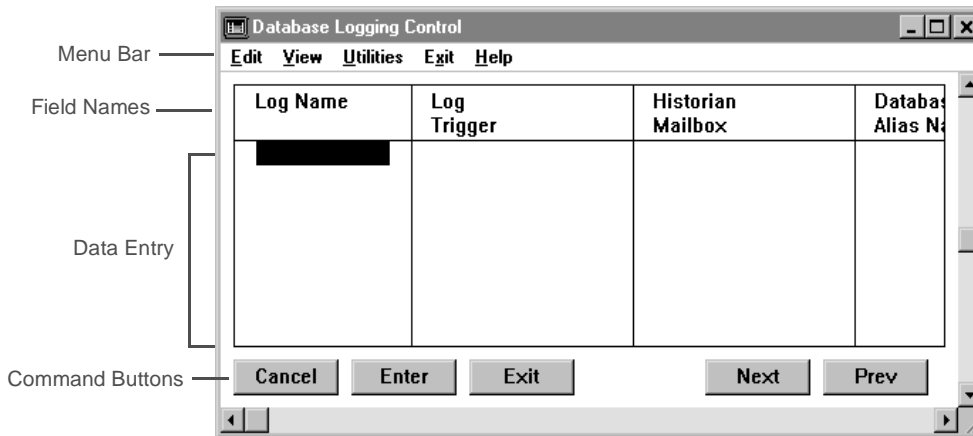
Text-entry configuration panels contain the following areas:

- **Menu bar**—The menu bar provides access to a group of tools you can use to manipulate or review configuration and application data.
- **Data entry**—Type ASCII text in the data entry area.

WORKING WITH STRUCTURED CONFIGURATION PANELS

You open configuration tables by choosing a module from the Main Menu. Data is entered in configuration tables through either structured panels or text-entry panels. This section discusses structured panels and their characteristics.

Configuration tables are typically built using structured panels. Each table has one or more panels. Each panel contains a number of data entry fields formatted in columns.



Structured configuration panels contain the following areas:

- **Menu bar**—The menu bar provides access to a group of tools you can use to manipulate or review configuration and application data.
- **Field names**—This area lists the names of fields pertaining to the configuration table.
- **Data entry**—You type data for each table entry in this area. Use the following keypad keys to move around in the data entry section.
 - Press TAB to move forward from one field to the next. If the current field is the last field in a row, the cursor moves to the first field of the next row.
 - Press SHIFT + TAB to move backward from one field to the next. If the current field is the first field in a row, the cursor moves to the last field of the previous row.
 - Use the ↑ and ↓ keys to move up or down within a column in a panel.

You can optionally use a mouse to choose a field.

- **USING THE CONFIGURATION MANAGER**

- *Working with Structured Configuration Panels*

-
-

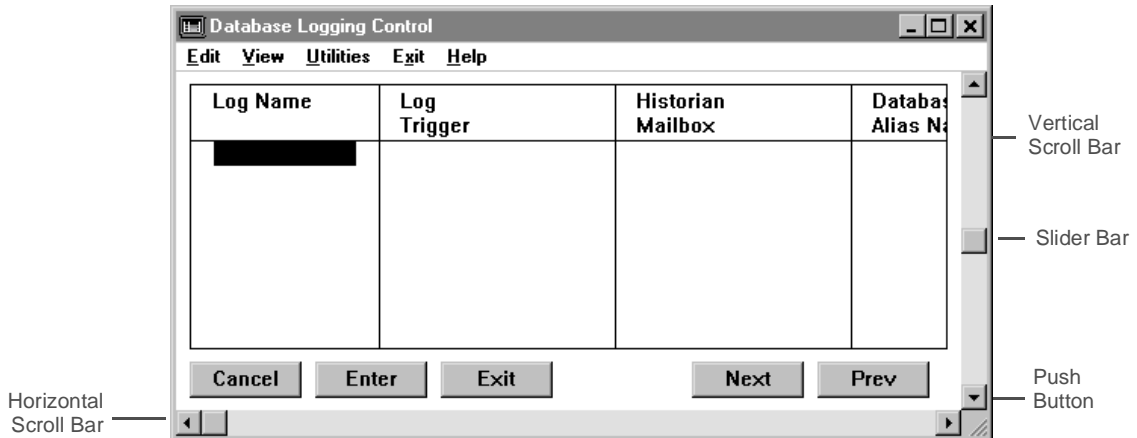
There are two ways to update information entered on this panel to the configuration table.

- Press the ENTER key on the keyboard. All line entries are updated to the configuration table. Press the TAB key at the end of each line rather than ENTER if you want to fill out the entire table before adding the data to the configuration table.
- Choose the Enter command button. All line entries are updated to the configuration table.
- **Command buttons**—Buttons that execute commands when you choose them.
 - **Cancel**—Cancels any data entered that has not already been updated to the configuration table.
 - **Enter**—Updates the configuration table with any data entered since the last update.
 - **Exit**—Returns you to the Main Menu. If you have not saved changes, FactoryLink asks you if you want to save or discard.
 - **Next**—Displays the next panel in a sequence of panels. If the current panel displayed is the last panel in the sequence, the first panel is displayed.
 - **Prev**—Displays the previous panel in a sequence of panels. If the current panel displayed is the first panel in the sequence, the last panel is displayed.

SCROLLING A WINDOW OR PANEL

If a window or panel contains more rows or columns than can be displayed in the available area, it contains a scroll bar which indicates more data is available for viewing.

A window or panel can contain both or either a vertical and horizontal scroll bar.



Scroll bars end in small buttons with arrows, called push buttons. Use these buttons to move the data in the window up and down one line at a time, or left and right one column at a time.

A scroll bar also contains a rectangle called a slider box. The slider box indicates the area of data to display. You can drag the slider anywhere along the horizontal scroll bar.

Vertical Scroll Bar

Use the vertical scroll bar to move the window's contents up and down in two ways.

- Point to the pushbutton at either end of the scroll bar and click the left button to move up or down one line.
- Position the pointer above or below the slider box and click the left button to move up or down one page.

- **USING THE CONFIGURATION MANAGER**

- *Scrolling a Window or Panel*

-
-

This scroll bar does not reflect the relative position in a panel and its slider box remains in a fixed position. Do not attempt to select this slider box and move it with the mouse. The results are unpredictable.

Horizontal Scroll Bar

Use the horizontal scroll bar to move the window's contents to the right or to the left in three ways:

- Point to the push button at either end of the scroll bar and click the left button to move to the left or right one column at a time.
- Choose the slider box and press and hold the left button while moving slider box along the vertical scroll bar to scroll horizontally. Release the left button when you finish.
- Point to the left or right of the slider box and click the left button to jump left or right.

WORKING WITH MULTIPLE DEVELOPMENT APPLICATIONS

You can use the Configuration Manager to develop any number of applications simultaneously; however, each application requires a unique directory for storing configuration tables. For example, assume you are building two applications: one for collecting data from a station in the Southern part of town and another for collecting data from a station in the Northern area. Both applications are located on the C: drive. The path to the Southern development application is \station\south; the path to the Northern development application is \station\north.

The following sections describe how to open different applications and share information between them.

Opening a New Development Application

Although you can develop multiple FactoryLink applications simultaneously, you can actively work on only one at a time. When you open the Configuration Manager, the application located in the default FLAPP directory is active. Perform the following steps to open a development application other than the default application.

- 1 Choose Close from the Configuration Manager Application pull-down menu to close the active application. Only one application can be open at a time.



- 2 Choose Open from the Application menu. You are prompted for the directory containing the application you wish to open.



- 3 Enter the full pathname of the directory containing the application you want to open.

- **USING THE CONFIGURATION MANAGER**
- *Working with Multiple Development Applications*
-
-

4 Choose the command button of the operation you want to perform.

- | | |
|--------|---|
| Enter | To open the application. For example, if you want to open the Southern application, enter C:\station\south. |
| Cancel | To return to the Configuration Manager without choosing an application. |
| Help | To get help. |

Sharing Information Between Applications

Often, two applications are similar. When this is the case, you can save development time by exporting database files from one application and importing them into another application. This section describes how to export and import application database files.

Exporting an Application

Exporting converts all database files into ASCII text files in preparation for being imported into another application. Once files are exported, they are saved in the FLAPP default directory. Tables in the shared domain are saved in the FLAPP/SHARED directory; tables in the user domain are saved in the FLAPP/USER directory. Each table generates its own file; each file name includes an .exp extension. The .exp files can be edited using an ASCII text editor; however, any editing error can cause cascading errors in a FactoryLink application, so use care when editing these files.

Perform the following steps to export database files from one application so they can be imported to another application.

- 1 Open the application you want to export. Refer to “Opening a New Development Application” on page 221 for details on opening an application.

- 2 Choose Utilities from the Configuration Manager or configuration panel menu bar. If you open it from the Configuration Manager, the entire application is exported. If you open it from a configuration panel, only that panel's information is exported.



The following pulldown menu is displayed.



- 3 Choose Export from the pulldown menu. The database files are exported.

Importing an Application

Importing converts ASCII text files created using the export function to database files in a new application. If a file with the same name already exists in the application when you import a file, the contents of the imported file are placed at the end of the existing file. Use care when doing this as no merging occurs. Duplicate records may be inserted.

Perform the following steps to import database files to an application.

- 1 Export the database files for the application you want to import to a new application. Refer to "Exporting an Application" on page 222 for details on how to do this.
- 2 Open the application where you want to import the exported application data. Refer to "Opening a New Development Application" on page 221 for details on opening an application.

- **USING THE CONFIGURATION MANAGER**
- *Working with Multiple Development Applications*
-
-

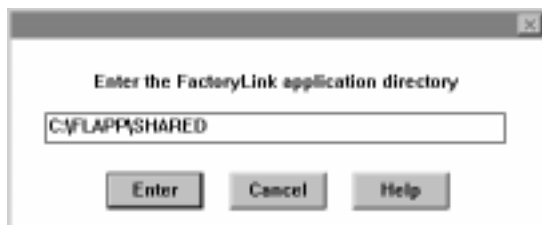
- 3 Choose Utilities from the Main Menu or configuration panel menu bar. If you choose it from the Main Menu, the entire application is imported. If you choose it from a configuration panel, only that panel's information is imported.



The following pulldown menu is displayed.



- 4 Choose Import. The following dialog is displayed requesting the name of the file you want to import.



- 5 Enter the full pathname of the directory containing the file(s) you want to import.

OPENING MULTIPLE CONFIGURATION TABLES

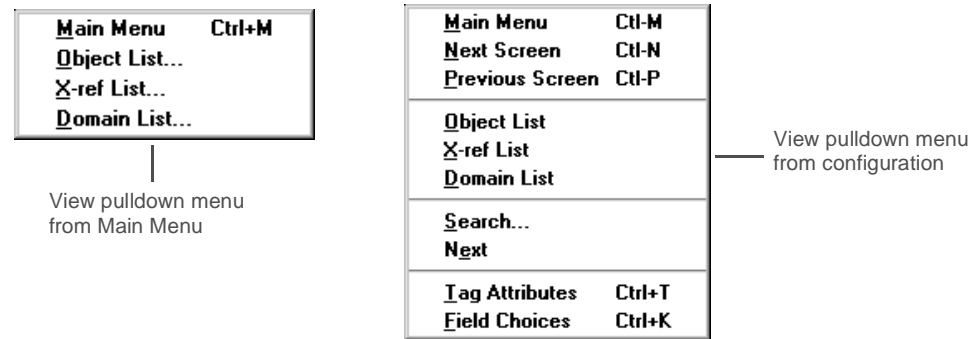
When you are configuring an application, sometimes you must fill out configuration tables for multiple modules. For example, if you are configuring logging of data from the real-time database to a relational database, you must define tables in the Historian, Schema, and Logger modules.

It can be beneficial in these cases to open configuration tables for multiple modules at the same time. Perform the following steps to open multiple configuration tables.

- 1 Choose View from the Main Menu or configuration panel menu bar.



One of the following pulldown menus is displayed.



- 2 Choose Main Menu. This makes the Main Menu the active panel.
- 3 Choose the module you want to open. This displays the panels comprising the configuration table for the selected module.

- **USING THE CONFIGURATION MANAGER**
- *Working with Tags*
-
-

WORKING WITH TAGS

This section describes how to manipulate tags defined in the application.

Viewing the Number of Tags Defined

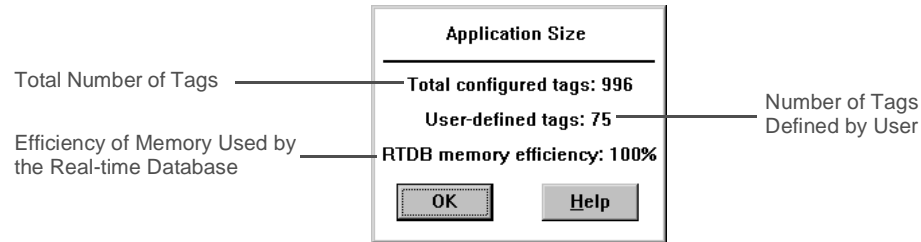
- 1 Choose Application from the Main Menu menu bar.



The following pulldown menu is displayed.



- 2 Choose Size to display information about the number of tags defined in the FactoryLink application. The following screen is displayed.



- 3 Choose OK to cancel this screen.

As the memory used by the real-time database becomes less efficient, the efficiency percentage drops, indicating wasted space. You can improve this by running the following command.

```
ctgen -r -c
```

Viewing a List of Real-time Database Elements

When a tag referencing a real-time database element is created in the application, it is added to an element list. Perform the following steps to view the real-time database elements configured for this application.

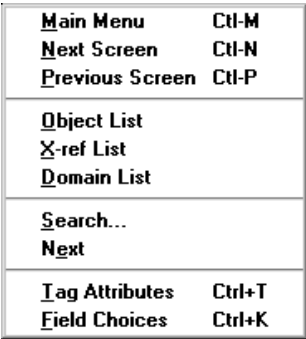
- 1 Choose View from the Main Menu bar or from the configuration panel menu bar.



One of the following pulldown menus is displayed:

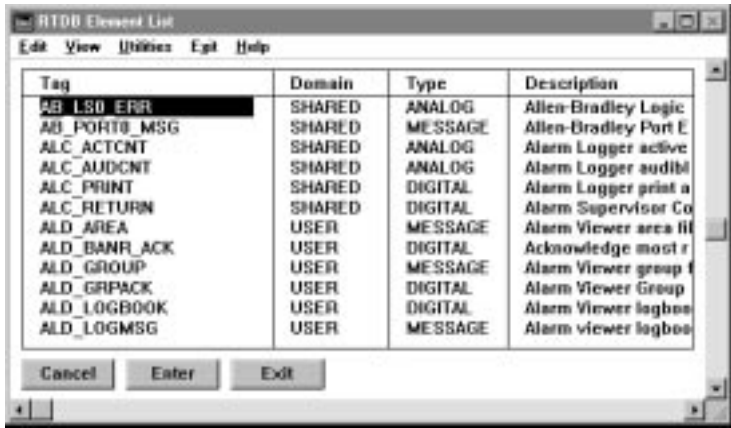


View pulldown menu from Main Menu



View pulldown menu from configuration

- 2 Choose Object List. The following list is displayed.



- **USING THE CONFIGURATION MANAGER**

- *Working with Tags*

-
-

This list contains an entry for each tag defined in the application. Tags are listed in alphabetical order. The list contains the following information for each entry.

| | |
|-------------|---|
| Tag | Logical name assigned to reference the real-time database element. |
| Domain | Domain in which the element is configured. This can be either shared or user. |
| Type | Data type of the element. Refer to “Defining Tags” in the <i>FactoryLink Configuration Guide</i> for a complete list of data types. |
| Description | Description of the purpose of the element. You can modify the entry in this field. |
| Task | Unused. |
| Value | Start up value for the tag. You can modify the entry in this field. |
| Seg. | Segment number where element is stored in the real-time database. FactoryLink uses the entry in this field in combination with the entry in the Offs. field to determine the location of the element. |
| Offs. | Offset number assigned to the element by FactoryLink. FactoryLink uses the entry in this field in combination with the entry in the Seg. field to determine the location of the element. |
| Size | Unused. |
| Flags | This field is currently not used. |
| Dimension | Array dimension size, if this entry is part of an array. Sizes are 1-based and are separated by commas. If the element is not an array, this field is blank. |
| Msg LEN. | Used only for message elements, indicates maximum number of characters permitted in the message. You can modify the entry in this field. |
| Persistence | Type of persistence assigned to this element. You can modify the entry in this field. Refer to the <i>Core Task Section</i> of the <i>FactoryLink Configuration Guide</i> for more information on the options. |
| Change Bits | Indicates whether or not an element’s change status flags are set to 1 (ON) or 0 (OFF) during a warm start. This column is blank if NONE is present in the Persistence field. You can modify the entry in this field. |

3 Choose Exit to exit this list.

Viewing Element Cross Reference List

A tag referencing a real-time database element can be used multiple times throughout the application. The element cross reference list defines each instance where the tag is used in the application. Perform the following steps to view where references to real-time database elements are displayed in the application.

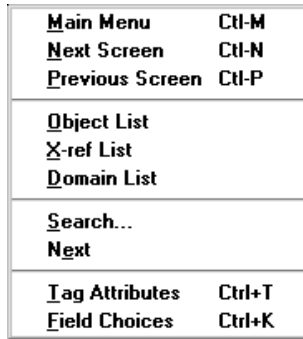
- 1 Choose View from the main menu bar or from the configuration panel menu bar.



One of the following pulldown menus is displayed.



View pulldown menu from Main Menu



View pulldown menu from configuration

- 2 Choose X-ref List. The following list is displayed on the screen.

| Tag Name | Tag Description | CT Domain | Task |
|--------------|-----------------|-----------|------|
| AD_LSD_EDH | | SHARED | EDI |
| AB_PORTS_MSG | | SHARED | EDI |
| ALC_ACTCNT | | SHARED | DIST |
| ALC_AUDCNT | | SHARED | DIST |
| ALC_AUDCNT | | USER | GRA |
| ALC_PRINT | | SHARED | DIST |
| ALC_PRINT | | USER | GRA |
| ALC_PRINT | | USER | GRA |
| ALC_RETURN | | USER | GRA |
| ALD_AREA | | USER | DIST |
| ALD_AREA | | USER | GRA |
| ALD_BANK_ACK | | USER | DIST |
| ALD_BANK_ACK | | USER | GRA |
| ALD_GROUP | | USER | DIST |
| ALD_GROUP | | USER | GRA |

- **USING THE CONFIGURATION MANAGER**
- *Working with Tags*
-
-

This list contains an entry for each instance a tag is referenced in the application. The information displayed for each instance is described below.

| | |
|---------------------------|--|
| Tag Name | Logical name assigned to reference the element in the real-time database. Each element will be displayed in this list as many times as it is referenced. |
| Tag Dimension | Dimension of the tag if the tag is part of an array. |
| CT Domain | Domain where element is referenced. |
| Task Name | FactoryLink module using element. |
| CT Name/Anim Type | Panel name referencing element. If this is a drawing, type of animation applied to element. |
| Table Name/ Drawing | Table name referencing element. If this is a drawing, name of drawing referencing element. |
| Record/ Object Name | Record number referencing element. If this is a drawing, object name assigned to object referencing element. |
| Column Name/Anim Field | Column name referencing the element. If this is a drawing, name of field in Animation dialog referencing element. |
| Sub-record/ Pen Name | Sub-record referencing element. If animation type is Pen, name of pen referencing element. |

3 Select Exit to exit this list.

Searching for a Tag

Typically there are a large number of tags defined in an application. Therefore, a mechanism for searching for a tag by name is provided. Use the following procedure to search for a specific tag in either the element list or element cross reference list.

- 1 Choose View from the main menu bar or from the configuration panel menu bar.



One of the following pulldown menus is displayed.

Main MenuCtrl+M

Object List...

X-ref List...

Domain List...

View pulldown menu from Main Menu

Main MenuCtrl-M

Next ScreenCtrl-N

Previous ScreenCtrl-P

Object List

X-ref List

Domain List

Search...

Next

Tag AttributesCtrl+T

Field ChoicesCtrl+K

View pulldown menu from configuration

- 2 Choose either Object List or X-ref List, depending on the kind of information you want to obtain about the element. For example, choose Object List to display the following element list.

105

Using the
Configuration
Manager

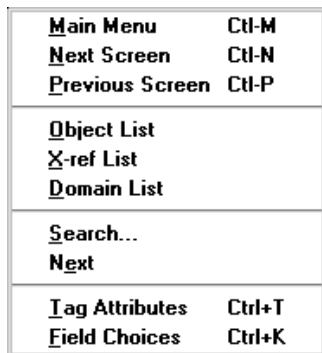
FactoryLink 6.5.0 / Fundamentals / 231

- **USING THE CONFIGURATION MANAGER**

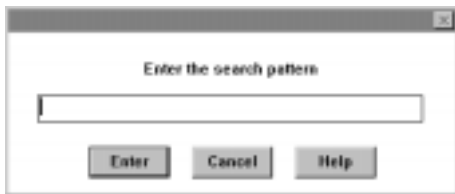
- *Working with Tags*

-
-

- 3 Choose View from the configuration panel menu bar. The following pulldown menu is displayed.



- 4 Choose Search. You are prompted for the name of the tag you want to find.



- 5 Enter the name of the tag you want to find and choose Enter. You can use an asterisk (*) in the name as a wildcard.
- 6 Choose Next from the View pulldown menu if you want to find the next occurrence of the search pattern .

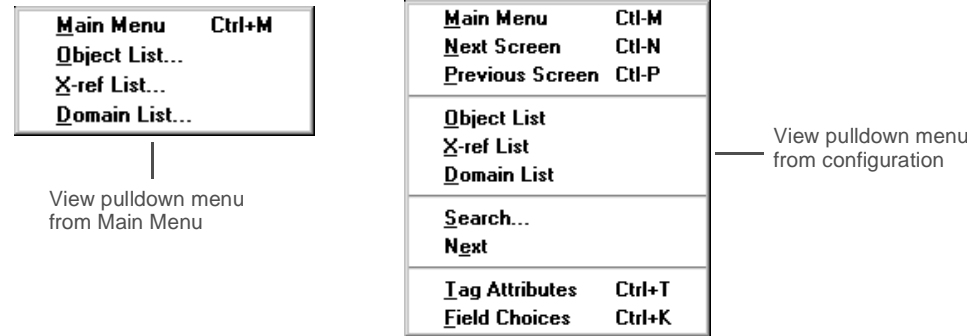
Deleting a Tag Definition

When you delete a tag from a graphics picture or configuration panel, the tag definition is not deleted from the element list. Perform the following steps to delete a tag definition.

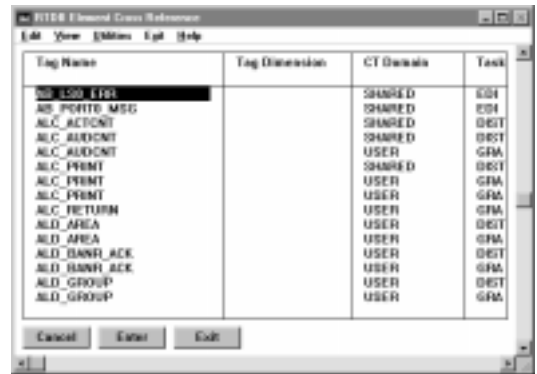
- 1 Choose View from the main menu bar or from the configuration panel menu bar.



One of the following pulldown menus is displayed.



- 2 Choose X-ref List. The following list is displayed on the screen.



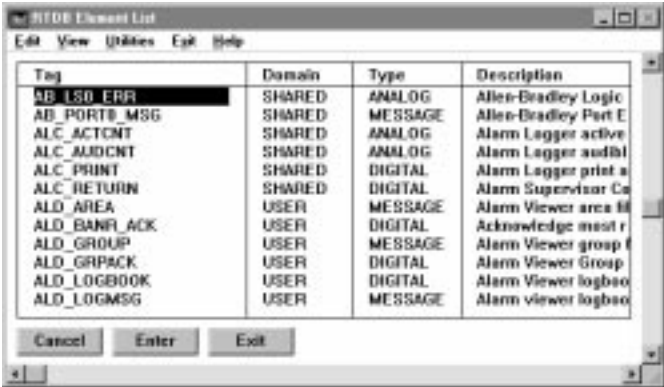
- 3 Check this list to be sure the tag you are deleting is not used anywhere else in the application. If it is, you will not be able to delete the definition from the elements list until each reference is deleted.

- **USING THE CONFIGURATION MANAGER**

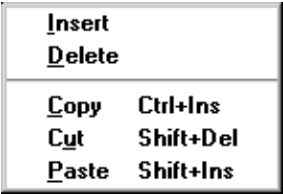
- *Working with Tags*

-
-

- 4 Choose Object List from the View pulldown menu once all references to the tag are removed.



- 5 Position the cursor on the line representing the tag you want to delete.
- 6 Choose File from the configuration panel menu bar. The following pulldown menu is displayed.



- 7 Choose Delete from this pulldown menu. The tag is deleted from the application.

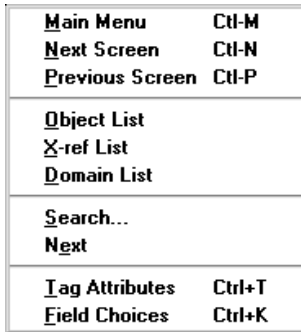
Changing a Tag Definition

Once a tag has been defined, perform the following steps to change its definition.

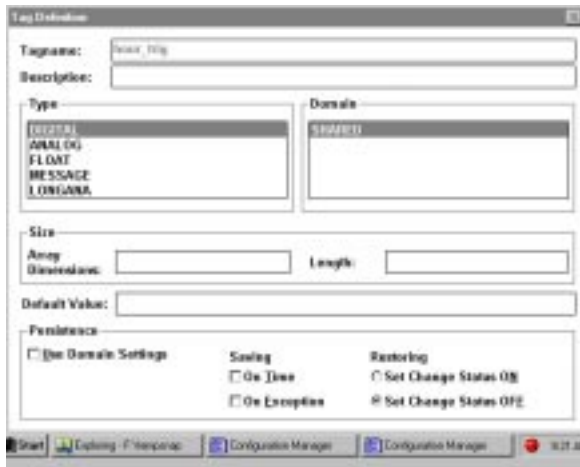
- 1 Position the cursor in a configuration panel field containing the tag name you want to edit.
- 2 Choose View from the Configuration Manager menu bar.



The following pulldown menu is displayed.



- 3 Choose Tag Attributes. The Tag Definition dialog is displayed.



- **USING THE CONFIGURATION MANAGER**

- *Viewing Miscellaneous Information*

-
-

- 4 Choose the Edit command button.
- 5 Edit the fields you want to change. Refer to “Defining Tags” in the *FactoryLink Configuration Guide* for descriptions of fields on this panel.

Or, you can press Ctrl+T to display this panel.

VIEWING MISCELLANEOUS INFORMATION

The Configuration Manager provides the following tools to view information about the FactoryLink application you are developing. Using these tools, you can view the following.

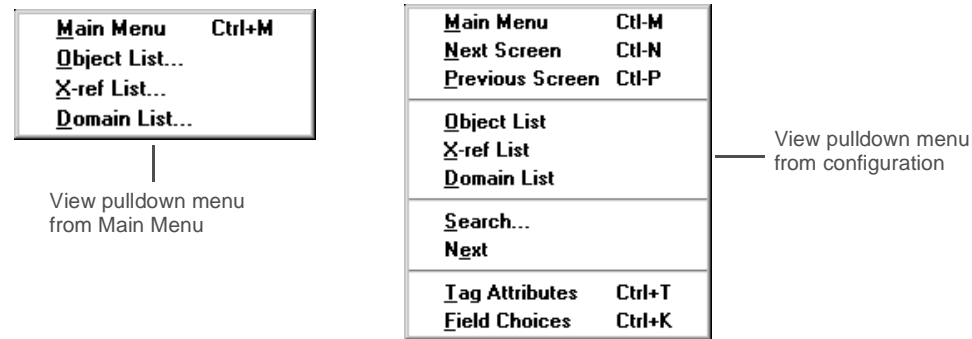
- Current version of FactoryLink
- Number of domains defined for an application

Viewing Domain List

- 1 Choose View from the main menu bar or from the configuration panel menu bar.



One of the following pulldown menus is displayed.



- 2 Choose Domain. The following list is displayed on the screen.

| DOMAIN | PARENT | # INST | Persistence | Change Bits |
|--------|--------|--------|-------------|-------------|
| SHARED | | 1 | NONE | OFF |
| USER | SHARED | 2 | NONE | OFF |

This list contains an entry for each domain defined. This will be one shared domain and one user domain. The information displayed for each domain is described below.

- Domain Domains configured for the application.
- Parent The parent of the domain. For user domain, this is shared. The shared domain does not have a parent.
- #INST Number of users that can open the domain at any one time. For shared domain, this defaults to 1 and should not be changed. For user domain, this defaults to 2 and should be changed to the maximum number of users you want to open the application at any one time.
- Persistence Type of persistence assigned to this element. You can modify the entry in this field. Refer to the *Core Task Section* of the *FactoryLink Configuration Guide* for more information on the options.
- Change Bits Indicates whether or not an element's change status flags are set to 1 (ON) or 0 (OFF) during a warm start. This column is blank if NONE is present in the Persistence field. You can modify the entry in this field.

- 3 Choose Exit to exit this panel.

- **USING THE CONFIGURATION MANAGER**

- *Viewing Miscellaneous Information*

-
-

Viewing the Current Version of FactoryLink

- 1 Choose Help from the main menu bar or configuration panel menu bar.



The following pulldown menu is displayed.



- 2 Choose About to display the current version of FactoryLink. The following screen is displayed.



- 3 Choose Enter to cancel this screen.

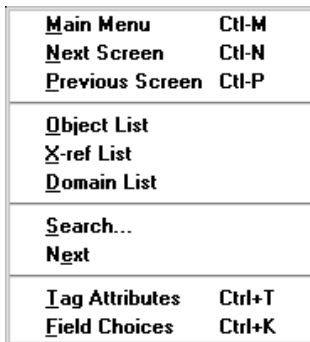
Viewing Field Choices

When you are filling out a field on the configuration panel that has a predetermined set of valid entries, you can view those entries using the following procedure.

- 1 Position the cursor in a configuration panel field that has a predefined set of valid entries.
- 2 Choose View from the Configuration Manager menu bar.



The following pulldown menu is displayed.



- 3 Choose Field Choices. This displays a list of your choices.



- 4 Choose your option and press Esc. Your choice is updated to the field. Or, you can press Ctrl+K to display this list.

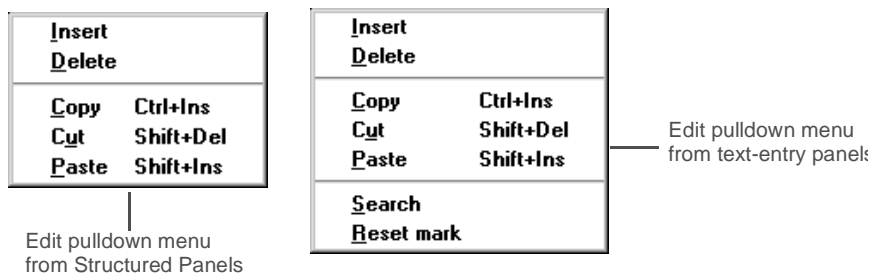
- **USING THE CONFIGURATION MANAGER**

- *Editing Configuration Panels*

-
-

EDITING CONFIGURATION PANELS

FactoryLink provides some editing tools to aid in editing configuration panels. Access these tools by choosing Edit from the configuration panel menu bar. One of the following pulldown menus is displayed.



The following sections describe the available editing functions.

Inserting a Blank Line

- 1 Position the cursor on the line below where you want the blank line.
- 2 Choose Insert from the Edit pulldown menu.

A blank line is inserted above the current line.

Deleting a Line Entry

- 1 Position the cursor on the line you want to delete.
- 2 Choose Delete from the Edit pulldown menu.

Before deleting the line, the system validates the display and reports any validation errors encountered. If none are encountered, it updates the real-time database.

- 3 Correct any validation errors and try again.

Caution: If you want to delete a line entry from a control panel that has information defined on the corresponding information panel, you must first delete the information panel definition. If you do not, the data defined on the information panel becomes orphaned and cannot be deleted, although it will still be displayed on the cross-reference list. This is also true if you change the table name on the control panel.

Copying Contents of a Line Entry

- 1 Position the cursor in the first line of the block to be copied.
- 2 Choose Copy from the Edit pulldown menu.
- 3 Move the cursor to the last line in the block to be copied. The selected block is highlighted.
- 4 Choose Copy from the Edit pulldown menu. The selected block is copied to the Paste buffer.

Choose Reset Mark from the Edit pulldown menu to cancel the copy operation in a text-entry panel .

Cutting a Line Entry

- 1 Position the cursor in the first line of the block to cut.
- 2 Choose Cut from the Edit pulldown menu.
- 3 Move the cursor to the last line in the block to cut. The selected block becomes highlighted.
- 4 Choose Cut from the Edit pulldown menu. The selected block is cut from the panel and copied to the paste buffer.

Choose Reset Mark from the Edit pulldown menu to cancel the cut operation in a text-entry panel.

- **USING THE CONFIGURATION MANAGER**

- *Editing Configuration Panels*

-
-

Although the cut option resembles the delete option in that the selected data is removed from the panel, the delete option does not move the data to the paste buffer. Unlike material that has been cut, deleted material cannot be reinserted in the panel.

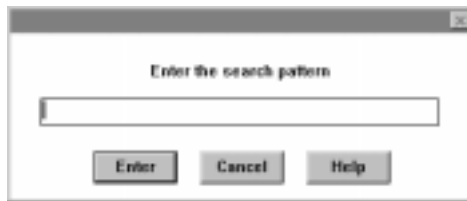
Pasting a Copied or Cut Line Entry

- 1 Perform a copy or cut operation as described above.
- 2 Position the cursor where you want the copied or cut data inserted.
- 3 Choose Paste from the Edit pulldown menu.

You can paste the block more than once because the Paste operation does not clear the paste buffer.

Searching for a Text String

- 1 Choose Search from the Edit pulldown menu. The following panel is displayed.



- 2 Enter the text you want to find.
- 3 Choose the Enter button to start the search. FactoryLink locates the next occurrence of the text in the current file, starting at the cursor position and searching downward.

Repeat this action to search for any additional entries of the text.

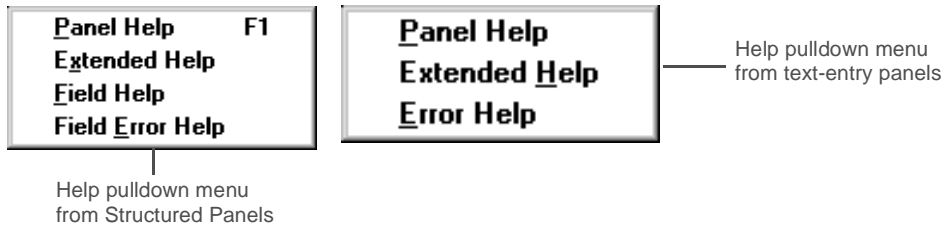
Choose Cancel to end the search and return to the Configuration Manager.

GETTING HELP

- 1 Choose Help from the main menu bar or configuration panel menu bar.



One of the following pulldown menus is displayed.



- 2 Choose one of the following, depending on the kind of help you want.

| | |
|------------------|--|
| Panel Help | Provides a brief explanation of the active configuration panel. |
| Extended Help | Provides general information about using the Configuration Manager. |
| Field Help | Provides a brief explanation of the current field. |
| Field Error Help | Provides help on errors. Provides information on any errors reported for this panel. |

- **USING THE CONFIGURATION MANAGER**

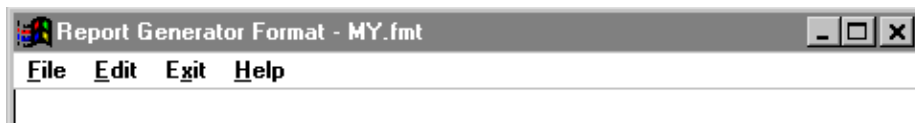
- *Managing Text-entry Panel Files*

-
-

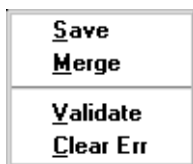
MANAGING TEXT-ENTRY PANEL FILES

FactoryLink provides tools to aid in managing the files created using text-entry panels, such as in Math & Logic or Report Generator. Perform the following steps to access these tools.

- 1 Choose File from the text-entry panel menu bar.



The following pulldown menu is displayed:



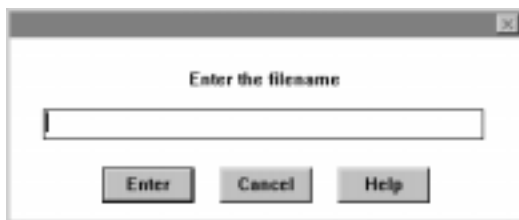
The following section describes the available file management functions.

Saving File Contents

Choose Save from the File pulldown menu to save the file as it is currently displayed. To avoid losing large amounts of data as the result of a power failure or other unexpected occurrence, save the file often. The Save operation does not validate the data.

Merging a File

- 1 Position the cursor on the line above the line where you want to insert the text file.
- 2 Choose Merge from the File pulldown menu. The following dialog is displayed.



- 3 Specify the name of the file to merge with the current file and press the Enter key.

Checking Syntax

Choose Validate from the File pulldown menu to check the current file for correct syntax. If invalid entries exist, an error message is displayed and the errors are highlighted.

Clearing Errors

Choose Clear Err from the File pulldown menu to clear all validation error flags from the current file. Clear Err does not remove or correct the errors. It simply removes the highlighting from the errors and makes it easier to read the display.

Changing Font Size

The default font for text-entry panels is Courier and the font size is 20. You can change the font size using the FLFONT environment variable. For example, to change the font size to 30, enter the following command at the system prompt.

```
set FLFONT=30
```

- **USING THE CONFIGURATION MANAGER**

- *Reporting*

-
-

REPORTING

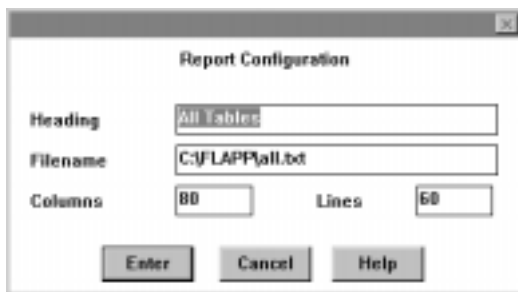
- 1 Choose Utilities from the main menu bar or from the configuration panel menu bar. If you choose it from the main menu bar, the report is generated for the entire application. If you choose it from a configuration panel menu bar, the report is generated for that panel.



The following pulldown menu is displayed:



- 2 Choose Report. The following dialog is displayed requesting information about how and where to generate the output.



- 3 Define the following fields.

- | | |
|----------|---|
| Heading | Name of the table to include in the report. Enter the name as it is displayed on the Main Menu. |
| Filename | Name of the file to receive the output. The default file name is FLAPP\ALL.TXT. |
| Columns | Width of the report in number of characters permitted on a single line. The default is 80 characters. |
| Lines | Length of the report in number of lines permitted on a single page. The default is 60. |

- 4 Choose Enter to process the report. FactoryLink converts the specified table to ASCII and sends it to the specified disk file. Or, choose Cancel to return to the Configuration Manager without generating a report.
- 5 View or print the report using any text editor. The report has a title page, a table of contents, and a section for each task configured in your application. A portion of a sample report is shown below. The sample shows the information provided for each task.

| System Configuration Information | | page | 1 |
|----------------------------------|---|------------------|---|
| Domain Name : SHARED | | | |
| <hr/> | | | |
| Flags | : | FSR | |
| Task Name | : | RUNMGR | |
| Description | : | Run-Time Manager | |
| Start Trigger | : | TASKSTART_S[0] | |
| Task Status | : | TASKSTATUS_S[0] | |
| Task Message | : | TASKMESSAGE_S[0] | |
| Display Status | : | TASKDSTATUS_S[0] | |
| Display Name | : | TASKNAME_S[0] | |
| Display Description | : | TASKDESC_S[0] | |
| Start Order | : | 0 | |
| Priority | : | 201 | |
| Executable File | : | bin/runmgr | |
| Application Directory: | | | |
| Program Directory | : | | |
| Program Arguments | : | | |

- **USING THE CONFIGURATION MANAGER**

- *Configuration Manager Messages*

-
-

CONFIGURATION MANAGER MESSAGES

If errors occur while you are configuring a table, the Configuration Manager Main Menu error messages are displayed on the screen. This section lists these messages, describes their cause, and provides suggested actions.

AC file doesn't exist for domain file

Cause: The FLINK /AC/DOMAIN.AC does not exist or cannot be opened. If it does not exist, FactoryLink installation may have not completed normally. If the file cannot be opened, it may already be opened by another process.

Action: Re-run the FactoryLink installation if the file does not exist.

AC file doesn't exist for tag file

Cause: The FLINK /AC/OBJECT.AC does not exist or cannot be opened. If it does not exist, FactoryLink installation may have not completed normally. If the file cannot be opened, it may already be opened by another process.

Action: Re-run the FactoryLink installation if the file does not exist.

AC file doesn't exist for type file

Cause: The FLINK /AC/TYPE.AC does not exist or cannot be opened. If it does not exist, FactoryLink installation may have not completed normally. If the file cannot be opened, it may already be opened by another process.

Action: Re-run the FactoryLink installation if the file does not exist.

AC file doesn't exist for XREF file

Cause: The FLINK /AC/XREF.AC does not exist or cannot be opened. If it does not exist, FactoryLink installation may have not completed normally. If the file cannot be opened, it may already be opened by another process.

Action: Re-run the FactoryLink installation if the file does not exist.

Can't copy, cut block in progress

Cause: A copy operation was attempted while a cut block was in progress.

Action: Complete the cut or cancel it with the Esc key or the CANCEL button.

Can't create database file

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred or your hard drive is full.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Can't create edit window

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

- **USING THE CONFIGURATION MANAGER**

- *Configuration Manager Messages*

-
-

Can't create help dialog

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Can't Cut, Copy Block in progress

Cause: A cut operation was attempted while a copy block was in progress.

Action: Complete the copy or cancel it with Esc or Cancel.

Can't cut/copy without an enter

Cause: A cut or copy operation was attempted, but Enter key was not pressed.

Action: Complete the operation by pressing the Enter key or cancel it with Esc or Cancel.

Can't delete database rows without an enter

Cause: The current panel contains invalid information. The system will not function further until the validation errors are removed.

Action: Choose Cancel to correct the validation errors, or press the Esc key to erase all input since the last validation, .

Can't delete from read-only table

Cause: A delete operation was attempted on a read-only table.

Action: Do not attempt to perform a delete operation.

Can't delete, block operation in progress

Cause: A delete operation was attempted while a block operation was in progress.

Action: Complete the deletion or cancel it with Esc or Cancel.

Can't execute

Cause: The specified file does not exist or has been damaged.

Action: Re-install FactoryLink.

Can't find AC

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Can't find tag to be deleted from x-ref database

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

- **USING THE CONFIGURATION MANAGER**

- *Configuration Manager Messages*

-
-

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Can't find window

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Can't insert at this point in database file

Cause: You tried to insert a row in a configuration panel between consecutively numbered rows; rows cannot be inserted between consecutively numbered rows.

Action: Insert the row either before or after the two consecutive rows or resequence the panel. To resequence the panel, use the Configuration Manager Cut function to cut the consecutively numbered rows from the panel. Then, use the Paste function to reinsert the rows. Now, you can insert a row between the two rows. You can also resequence all the rows of data you can view in the panel at one time, or even resequence all of the data in the panel.

Can't insert into read-only table

Cause: An insert operation was attempted on a read-only table.

Action: Do not attempt to perform an insert operation.

Can't insert/delete while cut/copy is in progress

- Cause:** An insert or delete operation was attempted while a cut or copy operation was in progress.
- Action:** Complete the cut or copy operation or cancel it with the Esc key or Cancel box. Retry the insert or delete operation.

Can't load external validation module

- Cause:** The module may not exist in the specified path or may have been damaged.
- Action:** Re-install FactoryLink.

Can't locate options key

- Cause:** The option key may not be installed, or the license may not be enabled.
- Action:** Check that the key is installed. Verify the license is enabled by running the KEYINST and FLKEYVAL utilities.

Can't make frame window

- Cause:** Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.
- Action:** Verify the following:
1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
 2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
 3. All hardware is correctly set up and all of the hardware is compatible.
- Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

- **USING THE CONFIGURATION MANAGER**

- *Configuration Manager Messages*

-
-

Can't make subframe window

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Can't make task list window

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Can't open AC file

Cause: The specified AC file does not exist or cannot be opened. If it does not exist, FactoryLink installation may not have completed normally. If the file cannot be opened, it may be opened by another process.

Action: Re-run the FactoryLink installation if the file does not exist.

Can't open file

Cause: The system cannot open an output file. The disk may be full, or the path may not exist.

Action: Delete any unnecessary files if the disk is full. Create the path if it does not exist.

Can't open FL.OPT file

Cause: The FL.OPT file has become corrupt or been deleted. Or the FLINK environment variable is not set.

Action: Check the value of the FLINK environment variable. Also, verify the FL.OPT file exists in the directory FLINK/OPT.

Can't paste - paste buffer is empty

Cause: An attempt has been made to perform a paste operation, but the block to be pasted was not moved to the paste buffer.

Action: Move the block to be pasted to the paste buffer by performing a copy or cut operation. Then, retry the paste operation.

Can't paste at this point in database file

Cause: You tried to paste a row between consecutively numbered rows; you cannot insert rows between consecutively numbered rows.

Action: Paste the row either before or after the two consecutive rows or resequence the panel. To resequence the panel, use the Main Menu Cut function to cut the consecutively numbered rows from the panel. Then, use the Paste function to reinsert the rows. Now, you can insert a row between the two rows. You can also resequence all the rows of data you can view in the panel at one time, or even resequence all of the data in the panel.

- **USING THE CONFIGURATION MANAGER**

- *Configuration Manager Messages*

-
-

Can't process command line

Cause: The command line you entered to the Main Menu has an invalid argument.

Action: Enter a valid command line argument.

| Command | Description |
|----------------|--------------------|
|----------------|--------------------|

| | |
|-----------|-------------------------------------|
| -Pxxxxxxx | xxxxxxx = name of Program directory |
|-----------|-------------------------------------|

| | |
|-----------|---|
| -Axxxxxxx | xxxxxxx = name of Application directory |
|-----------|---|

| | |
|-----------|-----------------------------------|
| -Lxxxxxxx | xxxxxxx = name of Log device/file |
|-----------|-----------------------------------|

| | |
|-----------|---------------------------------|
| -Txxxxxxx | xxxxxxx = name of Program Title |
|-----------|---------------------------------|

Can't read domain database

Cause: The file DOMAIN.CDB is either missing or corrupted.

Action: Copy the DOMAIN.CDB file from another application. If you do not have another application, run FLNEW to create one. (However, make sure the new one is created in a different directory.) Refer to the *Windows NT and Windows 95 Installation Guide* for information about running FLNEW. After you copy the file, re-start the application.

Cause: The FLAPP is not set to a valid application.

Action: Set the FLAPP to a valid application.

Can't read options key

Cause: The option key file FL.OPT has become corrupt.

Action: Copy the file FLNEW.OPT from the installation diskette to the file FLINK/OPT/FL.OPT.

Can't read type database

Cause: The file DOMAIN.CDB is either missing or corrupted.

Action: Copy the DOMAIN.CDB file from another application. If you do not have another application, run FLNEW to create a one. (However, make sure the new one is created in a different

directory.) Refer to the *Windows NT and Windows 95 Installation Guide* for information about running FLNEW.

Cause: The FLAPP is not set to a valid application.

Action: Set the FLAPP to a valid application.

Can't update with null table name

Cause: No Table Name for this panel.

Action: Return to the Control panel and move the cursor to the line containing the correct table name. Return to the Information panel. The correct table name should now be displayed in the Table Name field.

Col number filename

Cause: The specified column contains an invalid entry.

Action: Open the column and correct the entry.

Configuration Manager must be closed before exiting Windows

Cause: You attempted to exit the Windows operating system before exiting the Main Menu.

Action: Exit the Main Menu before exiting Windows.

Couldn't create input window

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

- **USING THE CONFIGURATION MANAGER**

- *Configuration Manager Messages*

-
-

Couldn't get external module address

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Couldn't start text editor

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Database error

Cause: A .CDB or .CDX file is corrupt.

Action: Run the utility DBCHK to locate and repair corrupt .CDB and CDX files. For information about DBCHK, see Chapter 109, "FactoryLink Utilities," in this manual.

Cause: The operating system, third-party software, or hardware setup on your system is incorrect or incompatible.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Cause: An internal error has occurred.

Action: Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Database doesn't contain any printable records

Cause: A report or export operation was attempted on an empty database file.

Action: Do not attempt to export or report this database file.

Error allocating panel. Maximum = 12

Cause: Twelve panels are already created and displayed.

Action: Close some panels and then reopen the desired panel.

Error allocating table structure

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

- **USING THE CONFIGURATION MANAGER**

- *Configuration Manager Messages*

-
-

Error closing database

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Error opening database file

Cause: The specified “.CDB” database file does not exist. The user does not have sufficient privilege to access the file. The file is corrupted.

Action: Verify the following:

1. The existence of the database file.
2. The user has proper privilege to access the file.
3. The integrity of the database and index files using the DBCHK utility.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Error opening index file

Cause: The specified “.MDX” database file does not exist. The user does not have sufficient privilege to access the file. The file is corrupted.

Action: Verify the following:

1. The existence of the index file.
2. The user has proper privilege to access the file.
3. The integrity of the index file using the DBCHK utility.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Error selecting index

Cause: The specified index name does not exist. The file is corrupted.

Action: Verify the following:

1. The existence of the index file.
 2. The integrity of the index file using the DBCHK utility.
- Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Expected xxxxx

Cause: The .AC parser did not find what it expected.

Action: Fix the .AC and try again.

Expected:*expected record length*/Found:*actual record length*

Cause: The record read was longer than expected. The file being imported may not have come from an export of the same database. The wrong file may have entered.

Action: Ensure the correct file is being used.

External command line too long

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

External modules not supported at this time

Cause: This error is displayed only on FactoryLink for OS/2 or Windows. The Main Menu cannot load an external module. Either the module may not exist in the path or it may have been damaged.

- **USING THE CONFIGURATION MANAGER**

- *Configuration Manager Messages*

-
-

Action: Run CHKDSK. If there are no problems with the disk files, re-install FactoryLink.

Field not in key list

Cause: A string has been entered that is not in the key file (FLINK/KEY/xxx.KEY) specified in the .AC file.

Action: Modify the string to match, or add the string to the key file. The string must be in the same case as it is in the key file.

File has been modified since last save.

Select 'OK' to destroy changes or 'CANCEL' to return to the edit session.

Cause: An exit was attempted from a window in which the changes have not been written to the database.

Action: Choose OK to disregard the changes and exit the window .
Choose Cancel to return to the edit window.

File is too big

Cause: Not enough available memory to load the text file.

Action: Close any unnecessary windows or programs. Add memory to the system if this happens often.

***filename* not in key list**

Cause: A string has been entered that is not in the key file FLINK/KEY/xxx.KEY specified in the .AC file.

Action: Modify the string to match, or add the string to the key file. The string must be in the same case as it is in the key file.

Flag must be 'vbru'

Cause: A flag of other than v, b, r, or u was found in the .AC file.

Action: Delete the invalid flag.

Grace period has expired. Software must be registered

Cause: FactoryLink was installed without registering it within the 10 day grace period.

Action: Run FLKEYVAL and follow the instructions for registration.

I/O error while trying to access the domain file

Cause: The index or file may be damaged.

Action: Contact Customer Support.

I/O error while trying to access the tag file

Cause: The index or file may be damaged.

Action: Exit the Main Menu and try again. If this doesn't work, delete the index file (.MDX) and rebuild it.

I/O error while trying to access the XREF file

Cause: The index or file may be damaged.

Action: Contact Customer Support.

Internal error

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Invalid dimension definition

Cause: The screen contains an invalid definition for dimension.

Action: Move the cursor to this field and choose Error Help or press the Alt-E key sequence to find out the reason for the failure.

- **USING THE CONFIGURATION MANAGER**
- *Configuration Manager Messages*
-
-

Invalid dimension size

Cause: The screen contains an invalid definition for dimension.

Action: The specified dimension may exceed size limitations. A dimension cannot exceed 16 characters.

Invalid dimension syntax

Cause: The screen contains an invalid definition for dimension.

Action: Move the cursor to this field and choose Error Help or press the Alt-E key sequence to find out the reason for the failure .

Invalid fields

Cause: The screen contains invalid information.

Action: Move the cursor to this field and choose Error Help or press the Alt-E key sequence to find out the reason for the failure.

Invalid file name

Cause: The specified file name is missing, too long, or has the wrong format.

Action: Enter a valid file name.

Invalid tag or dimension string length

Cause: The screen contains an tag or dimension string that exceeds the allowable size.

Action: The specified item exceeds size limitations. An element name cannot exceed 32 characters and a dimension cannot exceed 16 characters (total 48 characters). Correct the item that exceeds the size limitations.

Missing KEY file

Cause: The system can't find the KEY file FLINK/KEY/xxx.KEY.

Action: Enter the correct key file name in AC or create the desired key file.

Multiple panels

Cause: There is more than one panel statement in the AC window definition.

Action: Delete unnecessary panel statements.

No dialog for file selection

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

No dialog for tag selection

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

- **USING THE CONFIGURATION MANAGER**

- *Configuration Manager Messages*

-
-

No error

Cause: The current error field contains a valid entry.

Action: No action required.

No help yet

Cause: A .hlp file was not present for this field.

Action: Verify the related .hlp file is present in the FLINK /MSG directory and then run MKHELP.EXE.

No names in TITLES file

Cause: The file FLINK /AC/TITLES does not contain any .AC file names. FactoryLink installation may not have completed normally.

Action: Re-run the FactoryLink installation.

No message queue

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

No panel is defined

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).

2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

No RAM for ...

Cause: There is no more available RAM.

Action: Close any unnecessary windows or programs. Add RAM to the system if this happens often.

No TITLES file

Cause: The specified .AC file does not exist or cannot be opened. If it does not exist, FactoryLink installation may not have completed normally. If the file cannot be opened, it may already be opened by another process.

Action: Re-run the FactoryLink installation if the file does not exist.

No window for tag selection

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

- **USING THE CONFIGURATION MANAGER**

- *Configuration Manager Messages*

-
-

No window system

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Non-numeric character in a numeric field

Cause: A non-numeric character has been entered in a numeric field. Only characters 0-9, +, -, and . may exist in a numeric field.

Action: Delete the non-numeric characters from the field.

(Number of rows) rows were not imported due to errors

Cause: Invalid information has been entered. As a result, the system did not import the rows that contained errors.

Action: Correct the errors and retry the operation.

Numeric field is greater than maximum

Cause: A numeric value has been specified for a field greater than the maximum in the AC.

Action: Specify a value smaller than the maximum.

Numeric field is less than minimum

Cause: The specified numeric value for a field is less than the minimum in the AC.

Action: Specify a value larger than the minimum.

Open edit sessions exist

Cause: An attempt has been made to close the Main Menu window while one or more Edit windows were still open. If the Main Menu window is closed while Edit windows are still open, data not written to the database will be lost.

Action: Choose OK to exit. Choose Cancel to abort the exit.

Out of RAM

Cause: No more memory is available.

Action: Close any unnecessary windows or programs. Add more virtual memory to the system if this error occurs often.

Records can't be deleted from this file

Cause: An attempt has been made to delete records from a read-only file.

Action: Do not attempt to delete records from this file.

Records can't be pasted to this file

Cause: An attempt has been made to paste records to a read-only file.

Action: Do not attempt to paste records to this file.

Required field is blank

Cause: A field requiring entry is blank.

Action: Enter something other than spaces in this field.

Run-time only license. Development option not installed

Cause: A task may not be loaded and/or configured because it is not in the option key.

Action: Check the option key contents by typing UKEY -l.

Software has not been enabled

Cause: An error has occurred with the fl.key file in the opt directory.

Action: 1. Run KEYINST and if an invalid or archaic agreement is found, re-enter the configuration information.
2. Run FLKEYVAL and follow the instructions for registration.

- **USING THE CONFIGURATION MANAGER**

- *Configuration Manager Messages*

-
-

Tag already exists, but is not defined in, or above, the current domain level

Cause: You tried to define or reference a user domain element in the shared domain.

Action: Assign the element a new name or delete all references to the existing element.

Task is not installed properly

Cause: A task may not be loaded and/or configured because it is not in the option key.

Action: Check the master and option key contents by typing UKEY -l.

The system couldn't open the object, xref, or type database

Cause: FLAPP is not set to a valid application directory.

Action: Set FLAPP to a valid application directory.

Cause: The index or file specified may be damaged.

Action: Exit the Main Menu and try again. If this doesn't work, run DBCHK to identify the corrupt index file. Then, delete the index file (.MDX and rebuild it. For information about DBCHK, see Chapter 109, "FactoryLink Utilities," in this manual.

The Tag name *element name* contains one or more invalid characters

Cause: The element name specified contains one or more invalid characters.

Action: Correct the element name. The first character of the element name must be alphabetic or one of the following characters: @, \$, _ . The remaining characters can be numeric also.

The Tag name *element name* is undefined

Cause: The element name specified does not exist. Either a wrong name for an element has been entered, or the specified element has not been defined.

Action: Change the string to the name of an existing element or define the element when the system requests a definition.

The Tag type is invalid

Cause: If you are trying to import a file using the Import function, this error can occur if one of the elements being imported has the same name but is of a different data type than one of the elements that already exists in the current application.

Action: Either delete the element(s) with the same name from the file to be imported or rename the element. Then, re-import the file.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Cause: If you are not attempting an Import operation, either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have The correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

There are no errors because the file has not been validated

Cause: The VALIDATE function has not yet been used on this file. The standard edit window may mean the Enter key has not yet been entered for this window. It may also mean no validation exists for this application.

Action: No action required.

Too many keys! Using only 100.

Cause: There are too many keys in the key file. The maximum number of keys allowed is 100.

Action: Remove enough keys from the key file so there will be only 100.

- **USING THE CONFIGURATION MANAGER**

- *Configuration Manager Messages*

-
-

Unknown error: *error number*

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Validation error! Select OK to continue exiting

Cause: The text file contains invalid lines.

Action: Choose OK to exit with validation errors. Choose Cancel to abort the exit.

Using System Configuration

FactoryLink sets up default run-time parameters in the System Configuration Information panel during FLNEW.

The default values displayed on the System Configuration Information panel (as set up within an application newly created by FLNEW) establish the following parameters for the run-time FactoryLink system:

- Tasks that start up when the application is running
- Tasks allowed to run as foreground tasks
- Order in which tasks start up and shut down
- Priority of each task
- Domain associated with each task

Use this panel if you want to make the following modifications:

- Change task settings, such as flags and program arguments
- Re-sequence the order in which tasks start up
- Change foreground and background task identification
- Add a new task to an existing application

- USING SYSTEM CONFIGURATION
-
-
-

Choose System Configuration from the Configuration Manager Main Menu. The System Configuration Information panel is displayed:

| Flags | Task Name | Description | Start Trigger |
|-------|------------|---|---------------|
| S | RUNMGR | Run-Time Manager | TASKSTART_S |
| F | PERDS51 | Persistence | TASKSTART_S |
| FR | SCALE | Linear Scaling and Densification | TASKSTART_S |
| FR | TIMER | Interval and Event Timer | TASKSTART_S |
| F | ML | Interpreted Math and Logic | TASKSTART_S |
| FR | AL_LOG | Distributed Alarm Logger | TASKSTART_S |
| F | RPT | Report Logger/Generator | TASKSTART_S |
| F | RCORE | Bridge Server and Load | TASKSTART_S |
| F | COUNTER | Programmable Counter | TASKSTART_S |
| F | EDH | External Device Interface | TASKSTART_S |
| FR | DDM_HST | History for DDM (H [H]) | TASKSTART_S |
| F | DBLOG | Database Logger | TASKSTART_S |
| FR | DPLDGRN | Data Point Logger | TASKSTART_S |
| F | SPCData | Power SPC Data Collection & Calculation | TASKSTART_S |
| F | SPOOL | Spooler | TASKSTART_S |
| F | IPWNET | Client/Server RTD Network Interface | TASKSTART_S |
| F | FLM_SERVER | File Manager Remote Server | TASKSTART_S |
| F | FLANDRD | Local Area Network Send | TASKSTART_S |
| F | FLANRCV | Local Area Network Receive | TASKSTART_S |
| FS | RTMON | Real-Time Database Monitor | TASKSTART_S |

Specify the following information:

Flags Process to be performed by the system. A task can have multiple flags with flag values entered in any order. The entries and descriptions are as follows:

- S Session flag - Provides the process with its own window. Any output to that process or task is directed to this window rather than to the Run-Time Manager window.
The RUNMGR, RTMON, and GRAPHICS processes require their own windows.
- R Run flag - Activates this task at FactoryLink startup.
To allow a task to be started manually by an operator, do not enter the R flag.
- F Foreground flag - Puts this task in the foreground at startup. Use the F flag if the task has neither the S nor the R flags.

- Task Name** Enter an alphanumeric string of between 1 to 32 characters to define the name of the process (task), such as RECIPE, ALOG, or TIMER. Do not modify these default names. For an external program, define a program name.
- Description** Enter an alphanumeric string of between 1 to 80 characters to define the description of the task listed in the Task Name field. For example, the description of the SPOOL task is "Print Spooler."

The next six elements contain information about the task at run time. This information is displayed on the Run-Time Manager screen.

- Start Trigger** Name of an element that triggers (provides the mechanism for starting) the task. If the Flags field contains an R for this task, at run time a 1 (ON) is written to this element. This causes FactoryLink to start the task automatically. If the Flags field for this task does not contain an R, the task does not start until the operator writes a 1 (ON) to this element by clicking on the task name on the Run-Time Manager screen.

The data type for this tag can be digital or analog depending on the kind of data stored in the tag. The default is digital.

- Task Status** Name of an element that contains the analog status of the task. This element can have the following analog values:

- 0 Inactive
- 1 Active
- 2 Error
- 3 Starting
- 4 Stopping

The data type for this tag is analog.

- Task Message** Name of an element to which the task listed in the Task Name field writes any run-time messages. These messages appear in the Message column of the Run-Time Manager screen.

This element can have the following message values:

- | | | |
|----------|----------|-------|
| Inactive | Active | Error |
| Starting | Stopping | |

- USING SYSTEM CONFIGURATION
-
-
-

The data type for this tag is message.

| | |
|---------------------|---|
| Display Status | Name of an element that contains a text version of the status of the process. The task status is displayed in the Status column on the Run-Time Manager screen. The data type for this tag is message. |
| Display Name | Name of an element that contains the string entered in the Task Name field. This task name is displayed in the Task column on the Run-Time Manager screen. The data type for this tag is message. |
| Display Description | Name of an element that contains the string entered in the Description field on this panel. The data type for this tag is message. |
| Start Order | Enter an alphanumeric string of between 0 to 31 characters to define the order in which tasks are started at run time. The task defined with Start Order 0 starts first. Start Order 1 starts next, and so on, until Start Order 31, which starts last. Tasks with the same start order number will start consecutively. The default Start Order for the Run-Time Manager task is 0. Use the following guidelines to determine the Start Order for certain tasks: |

| Set the Start Order for... | To start... |
|----------------------------|--|
| Historian | before Logger |
| Logger | before Trending |
| Math & Logic | coordinated with the functions defined in the procedures. For example, if a procedure is dependent on data from an external device, start EDI before Math & Logic. |

| | |
|-----------------|--|
| Priority | Processing priority for the task. The priority is a three-digit hexadecimal value which is divided into two parts: |
| Caution: | Unless you are knowledgeable and experienced in setting priorities in the operating system, you might decide to leave the priority at the default value. |

First part (the 2 in 201): Hex value that specifies the operating system class of priority as listed below. This part is inactive in Windows.

- 0 Current class unchanged
- 1 Idle
- 2 (default) Regular
- 3 Time-critical

Note: Do not set any FactoryLink task to priority class 0 or 1. Use caution in assigning a task with priority class 3 because a time-critical task takes priority over a foreground task. A foreground task takes priority over regular tasks at run time.

Second part (the 01 in 201): Two-digit hexadecimal value (00 to 1F [0 - 31 decimal]) that specifies the priority within the priority class listed above. The higher the number, the higher the priority within the class.

Hexadecimal value 00 to 1F (default = 01)

Refer to the appropriate operating system guide for programming for further information about processing priority within an operating system.

Third part (the 01 in 201): Two-digit hexadecimal value (00 to 1F [0 - 31 decimal]) that specifies the priority within the priority class listed above. The higher the number, the higher the priority within the class.

Hexadecimal value 00 to 1F (default = 01)

Refer to the appropriate operating system guide for programming for further information about processing priority within an operating system.

Executable File Location of the executable file. If this is a relative path name to FLINK, do not use leading spaces.

Any path name which has the following format:

\DIRECTORY\SUBDIRECTORY\FILENAME

For example, bin\iml

Note: Do not use a file extension of .exe for this entry. It is not required and can cause undesirable results if the application is ported to UNIX.

- USING SYSTEM CONFIGURATION
-
-
-

- Application Directory This field is reserved for future use.
- Program Directory This field is reserved for future use.
- Program Arguments Values used as the arguments to the process. If this field is blank, no arguments are passed to the task. Program arguments are not case-sensitive. The table below shows valid arguments for applicable tasks:

Table 106-1 Valid Program Arguments

| FactoryLink Task | Valid Entries | Descriptions |
|--|---------------|---|
| Data Alarm Logger | -T | Enables the limit value to be written to the limit tag. If the -T option is not specified, the limit value is not initialized upon Data Alarm Logger startup. |
| Batch Recipe | -L -L -V# | Log error messages to a file Log error messages with more information (increased verbose level) to a file |
| Database Browser | | |
| Historians | | |
| Math & Logic (Interpreted mode only) | | |
| SPC Log, SPC View, and SPR | | |
| Trending | | |
| # is 2, 3, or 4. The greater the number, the more information you will receive. + is any number between 100 and 2000. * is any number from 2 to 22. The greater the number, the more information you will receive. | | |

Table 106-1 Valid Program Arguments (Continued)

| FactoryLink Task | Valid Entries | Descriptions |
|---|----------------------------|--|
| Database Logger | -L -L -V# -Q+ | Log error messages to a file Log error messages with more information (increased verbose level) to a file Increase the size of the Logger's buffer |
| Run-Time Graphics | -o1 (letter o) -T -P | Performs redraw for static objects Uses object's animated value Performs redraw of symbol background |
| Print Spooler | -M | Send print requests (except for alarm logs and binary files) to the system print queue instead of directly to the printer. |
| FactoryLink Local Area Networking (FLLAN) | -L | Log FactoryLink FLLAN error messages to a file |
| | -L -D* | Log FactoryLink FLLAN error messages with more information (increased verbose level) to a file |
| | -X | Log underlying network software's error messages to a log file |
| | -R | Prevents FactoryLink from automatically setting FLLAN's Enable/Disable tag to 1 (ON). |
| <p># is 2, 3, or 4. The greater the number, the more information you will receive.</p> <p>+ is any number between 100 and 2000.</p> <p>* is any number from 2 to 22. The greater the number, the more information you will receive.</p> | | |

- USING SYSTEM CONFIGURATION
-
-
-

Table 106-1 Valid Program Arguments (Continued)

| FactoryLink Task | Valid Entries | Descriptions |
|--|---------------|--|
| File Manager | -L -L -D* | Log error messages to a file Log error messages with more information (increased verbose level) to a file |
| # is 2, 3, or 4. The greater the number, the more information you will receive. + is any number between 100 and 2000. * is any number from 2 to 22. The greater the number, the more information you will receive. | | |

EDITING TASK INFORMATION

The new application utility, FLNEW, establishes default values for tasks in the System Configuration Information panel and associates each task with a specific domain. These defaults and associations are based on evaluation of task performance and are recommended for most applications.

You may edit this panel to identify an external program to the system. Although you can make changes in some fields, it is better not to change any fields except Flags and Display Status.

Viewing Domain Associations

Complete the following steps to view the tasks associated with a specific domain:

- 1 Ensure the current domain selected is correct in the Configuration Manager Domain Selection box.
- 2 Choose System Configuration from the Main Menu.

double click on System Configuration to open System Configuration Information panel.
- 3 Change the domain specified in the Domain Selection box to see the tasks in the System Configuration Information panel associated with the other domain.

Refer to “Domains for Run-time Tasks” on page 33 for information about domain defaults.

Adding New Tasks

The Run-Time Manager uses pre-defined elements and array dimensions to automatically display items onscreen. These element names and array dimensions appear in the System Configuration Information panel for each task displayed on the Run-Time Manager screen. If you add another task to the Run-Time Manager screen, use the next available array dimension.

- **USING SYSTEM CONFIGURATION**
- *Editing Task Information*
-
-

Determining the Next Array Dimension

The next available array dimension is determined by a task's position in the display sequence. If you view the System Configuration table associated with the SHARED domain, you will find all element names associated with Run-Time Manager end with a dimension of [0], Persistence element names end with a dimension of [1], Timer elements end in [2], and so on. For example, the complete entry in the Task Status field for the Run-Time Manager task is TASKSTATUS_S[0].

If the information in a field is longer than the number of characters that fit in the allotted space on the screen, part of the entry will scroll out of sight, as shown below:

| If this is the complete data in the field: | Then This Is Displayed in the 16-Character Space: |
|--|---|
| Programmable_Counter[7] | Programmable_Cou |

To see characters that have scrolled out of sight, press the → and ← keys. The field scrolls to display the text. The bracketed number represents an array dimension.

Complete the following steps to add a task to the Run-Time Manager screen:

- 1 Choose the appropriate domain for the task on the Main Menu.
- 2 double click on System Configuration to open the System Configuration Information panel.
- 3 Starting under the last row of information in the System Configuration Information panel, add the required information about the new task to each field. Use the Copy and Paste functions to copy duplicate information, such as element names, from the previous row.
- 4 Review the previous task in the task list to determine its dimension. Assign the new task's element names the next available array dimension. If you used the Copy and Paste functions to copy information from an existing row, modify each array dimension to be the correct value.
- 5 Choose Enter to save the information when the panel is complete.

The next time you run the application, the new task and its related information is displayed at the bottom of the specified domain's Run-Time Manager screen.

CHANGING DOMAIN ASSOCIATIONS

If an application has special task requirements, you can change its domain associations. Use the Main Menu's Delete, Copy, and Paste options from the Edit menu as outlined below.

- 1 Ensure the current domain selected is correct in the Configuration Manager Domain Selection box.
- 2 Open the System Configuration Information panel from the Main Menu.
- 3 Choose the task information you want to move.
- 4 Use the Edit menu's Delete function to remove the task information.
- 5 Modify the row for the last task in the current domain to use the array dimensions of the deleted task to prevent a blank line from appearing on the Run-Time Manager screen.
- 6 Choose Enter to save the changes.
- 7 Ensure the current domain selected is correct in the Configuration Manager Domain Selection box.
- 8 Open the System Configuration Information panel.
- 9 Starting under the last row of information in the System Configuration Information panel, add the required information about the new task to each field. You can use the Main Menu's Copy and Paste functions to copy duplicate information, such as element names, from the previous row.
- 10 Review the previous task in the task list to determine its dimension. Assign the new task's element names the next available array dimension.
- 11 Choose Enter to save the information when the panel is complete.

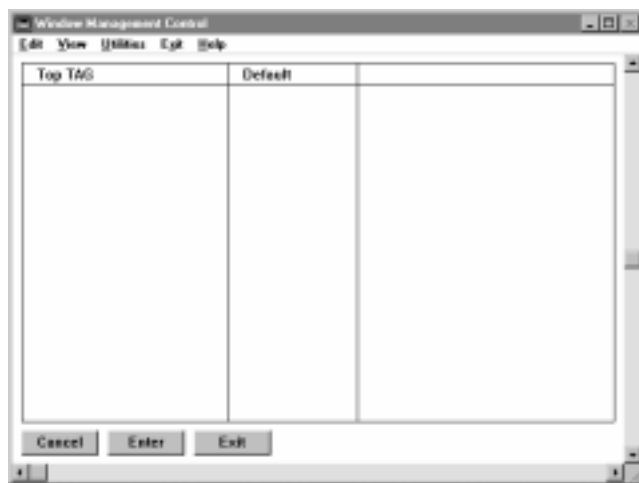
Note: You must add new items to the end of the panel to reflect the appropriate display sequence. See the previous section for more information about adding tasks to the System Configuration Information panel.

- **USING SYSTEM CONFIGURATION**
- *Setting the Run-Time Graphics Windows*
-
-

SETTING THE RUN-TIME GRAPHICS WINDOWS

You can determine which window should appear first (top) when the Run-Time Graphics task starts up. Use the Window Management Control panel to specify the top screen.

Choose the Windows Management option from the Main Menu. The Windows Manager Control panel is displayed with all fields visible:



Specify the following information:

- Top TAG** Name of the element used to determine the top window. The default is TOPWINDOW_U.
The valid data type is message.
- Default** Enter an alphanumeric string of between 1 to 8 characters to define the name of the default top window. This name must also be defined in the Application Editor window configuration. The default is FL_MAIN.

Note: New applications created with FLNEW use the default values.

CHANGING THE WINDOWS COLOR SCHEME

If some fields on configuration panels are difficult to read, you can edit the Windows color scheme to make them more legible.

Perform the following steps to do this:

Note: Configuration panels should be easy to read if you use the Windows default color scheme.

- 1 Choose Start>Settings>Control Panel to display the Control Panel dialog.
- 2 Choose Control Panel>Display to display the Display Properties dialog.
- 3 Click on the Appearance tab.
- 4 Choose the desired item from the Item: box. The Color: box displays the current color assigned to that item.
- 5 Click on the down arrow to the right of the displayed color to choose a new color from the Color: box. You might change the Menu, Selected Items, and Window Text colors so they contrast.
- 6 Click on Save As to save the changes and display the Save Scheme dialog with the current color scheme name.
- 7 Click on OK to save the changes to the current color scheme.

Or

Delete the current color scheme name and enter a new one to save the changes to another name. Then, choose OK to display the new color scheme.

- 8 Click on OK to exit the Display Properties dialog.

- **USING SYSTEM CONFIGURATION**

- *Archiving Error Messages*

-
-

ARCHIVING ERROR MESSAGES

This section contains information about setting up FactoryLink to create error message log files. Whenever an error occurs in a FactoryLink task at run time, FactoryLink sends a message for display to the Run-Time Manager screen. FactoryLink also sends a longer, more descriptive message to the log file (if the file has been set up). The following FactoryLink tasks can create an error message log file:

- Batch Recipe
- Database Logger
- Database Browser
- FactoryLink Local Area Networking (FLLAN)
- File Manager
- Any Historian
- Math & Logic (Interpreted mode only)
- Statistical Process Control (SPC Log, SPC View, and SPR)
- Trending

You can configure FactoryLink to automatically create an error message .LOG file at startup.

In Windows NT, .LOG files are stored in the FLAPP/FLNAME/FLDOMAIN/FLUSER/LOG directory.

In Windows 95, for tasks configured in the USER domain, the .LOG files are stored in the FLAPP/FLNAME/FLDOMAIN/FLUSER/LOG directory; for tasks in the SHARED domain, the .LOG files are stored in the FLAPP/FLNAME/FLDOMAIN/SHAREUSR/LOG directory.

where

FLAPP is the environment variable for the application directory.

FLNAME is the environment variable for the application name.

FLDOMAIN is the environment variable for the domain.

FLUSER is the environment variable for the user name.

FactoryLink creates the log file name using the following format:

XXMMDDYY.LOG

where

- XX indicates the FactoryLink task.
- MM is the month of the year (1-12).
- DD is the day of the month (1-31)
- YY is the year since 1900 (00-99).

If you specified during installation you wanted to install the Old version of FLLAN, FLLAN's .LOG files will have the following path and file names: FLAPP\NET\FLLANSND.LOG and FLAPP\NET\FLLANRCV.LOG

If you configure FactoryLink to create a log file for a task, FactoryLink logs a message in its log file whenever that task generates an error. The messages in the log file are more descriptive than those that appear on the Run-Time Manager screen.

For debugging purposes, configure FactoryLink to create log files automatically at startup. Complete the following steps to configure FactoryLink to do this:

- 1 Choose System Configuration from the Main Menu. The System Configuration Information panel is displayed.
- 2 Ensure the current domain selected is correct in the Configuration Manager Domain Selection box. Locate the corresponding entry for the task in the Task Name field.
- 3 Place the cursor on the corresponding entry.
- 4 Tab over to the Program Arguments field.
- 5 Enter -L, -V# (not case-sensitive) where # is 2, 3, or 4 in the Program Arguments field. The greater the number, the more information you receive. (Enter -L, -D# where # is any number from 2 to 22 for the File Manager and FLLAN tasks.)
- 6 Choose Enter to save the information.
- 7 Repeat steps 2 through 7 for each task that needs a log file.

- **USING SYSTEM CONFIGURATION**

- *Archiving Error Messages*

-
-

The log files continue to grow at run time as messages are logged to them until the operator shuts down and restarts each task. Then, FactoryLink creates new log files. However, FactoryLink creates only one log file per task per day no matter how many times each task is shut down and re-started in one day.

Delete old log files periodically to prevent log files from using too much disk space. You can configure the File Manager task to delete files for you. For example, File Manager can delete them each day at midnight or when the files specified reach a specified size.

Caution: Do not delete the current log file if the task is still running. This causes errors.

When you are finished debugging your application, you can remove the Program Arguments from the System Configuration Information panel to eliminate the creation of extra files.

RESIZING AND MOVING SCREEN COMPONENTS

You can modify and move the following FactoryLink screen components:

- Task panels
- Configuration Manager: the screen, the Main Menu, and the Domain Selection box
- Application Editor windows

The size and position of the screen components will return to their original default values when you exit.

Resizing a Screen

Complete the following steps to resize a screen component:

- 1 Position the cursor anywhere on the bounding borders. The cursor changes to a double-arrow.
- 2 Move the pointing device in either direction, as indicated by the arrow, to stretch or shrink the size.

Moving a Screen

Complete the following steps to move a screen component:

- 1 Position the cursor on the title bar. Press and hold the left button. If the file was inactive, it is now active.
- 2 Move the pointing device to the new location and release the left button.

- **USING SYSTEM CONFIGURATION**
- *Calculating the Number of FactoryLink Processes*
-
-

CALCULATING THE NUMBER OF FACTORYLINK PROCESSES

Use the following chart to calculate the processes required by FactoryLink. The total number of processes used by FactoryLink is the number of USER domain processes multiplied by the number of users plus the number of SHARED domain processes.

Table 106-2

| FactoryLink Processes | | |
|--|------------------------------------|--------------------------------------|
| FactoryLink IV Task | # of Processes for the USER Domain | # of Processes for the SHARED Domain |
| Run-Time Manager | 1 | 1 |
| Programmable Counters | 1 | 1 |
| Event & Interval Timers | NA | 1 |
| Interpreted Math & Logic | 1 | 1 |
| Alarm Supervisor | 1 | 1 |
| Application Editor | 1 | NA |
| Real-Time & Historical Trending | 1 | NA |
| Statistical Process Control: SPCVIEW SPR SPCLOG | *2 *2 NA | NA NA *2 |
| Print Spooler | NA | 1 |
| File Manager: CLIENT SERVER | 2 NA | NA 2 |
| Historian | NA | *2 |
| Database Browser | *2 | NA |
| Database Logger | NA | 1 |

Table 106-3

| FactoryLink Processes | | |
|--|------------------------------------|--------------------------------------|
| FactoryLink IV Task | # of Processes for the USER Domain | # of Processes for the SHARED Domain |
| Run-Time Monitor | 1 | NA |
| Report Generator | NA | 1 |
| Batch Recipe | NA | 1 |
| Local Area Networking: FLLANRCV FLLANSND | NA | 1 1 |
| External Device Interface | NA | 1+1 per protocol module |
| *Unless you are using INGRES, SYBASE, or dBASE IV, only one process is used. If you are using ORACLE, one of the processes counts as an ORACLE user license. NA indicates the USER or SHARED FactoryLink task does not need a process. | | |

- **USING SYSTEM CONFIGURATION**
- *Calculating the Number of FactoryLink Processes*
-
-

Using Run-Time Manager

Run-Time Manager is the user interface to the run-time environment. You run the application using the Run-Time Manager which allows you to start, monitor, and stop individual FactoryLink tasks.

This chapter describes how to open and use the Run-Time Manager.

SETTING UP PROGRAM ARGUMENTS

Tuning Kernel Memory

FactoryLink mailbox messages enable tasks to communicate between themselves via a queue as opposed to the standard FactoryLink single-value/change notification system. The queue removes the chance of the receiving task missing data should changes occur rapidly. Since all writes to a mailbox are stored in the kernel until read, however, kernel memory resources can be exhausted when the messages written into the kernel are not read by their target consumer task. Once these resources are exhausted, operations requiring mailbox communication, such as screen changes or database logging, no longer function.

Once a mailbox has been stuffed with orphaned messages, no other mailbox writes can be performed, even if these writes are to a different mailbox.

The system must be able to be tuned to handle large quantities of mailbox messages as well as not allow any mismatched mailbox producer/consumer task combinations exhaust the kernel of all resources.

Currently, the kernel defaults to the following configuration:

- 200 maximum memory segments of 64K bytes each
- 50 of these 200 segments can be used for mailbox messages
- One mailbox can retain as much memory as the overall maximum allows

- **USING RUN-TIME MANAGER**

- *Setting Up Program Arguments*

-
-

To make this configuration tunable, a switch to the SHARED domain instance of the Run-Time Manager is used. The syntax of this switch is

`-m<max_seg>[<max_mbxsegs>[:<max_onembx>]]`

where

| | |
|----------------------------------|--|
| <code>max_seg</code> | Maximum number of kernel segments. Default = 200. |
| <code><max_mbxsegs></code> | Maximum segments used for mailbox messages. Default = 50. |
| <code><max_onembx></code> | Maximum number of K bytes of message space held by one mailbox. The default sets no per-mailbox ceiling. |

This switch is configured through the Configuration Manager in the System Configuration Information panel.

Without the `-m` switch, the system uses the existing defaults of 200 total segments, 50 segments available to mailboxes, and no per-mailbox byte usage limit.

In addition to system-wide limits, a memory usage ceiling can be set per mailbox tag. The message length field of the Object table, currently supported for message tags, can be set with a maximum memory usage for its associated mailbox tag. This is specified in K bytes. The per-mailbox limit supersedes the system-wide mailbox limit.

Once a mailbox tag ceiling is reached, all subsequent writes to that tag are dropped. A new error code, `FLE_MBXLIM_EXCEED`, is returned for this case.

The per-mailbox K byte limit can also be set or obtained through the Programmer's Access Kit (PAK) task.

Additional Program Argument Options

Other program argument options include:

- `-f1`
- `-b`
- `-d`
- `-t`
- `-l`

See the following table for details of these arguments.

SETTING COMMAND LINE OPTIONS

There are several run-time options you can use to control how the tasks running in Run-Time Manager execute. These options are listed in the following table.

| Option | Action |
|---------------------|--|
| -d | Turns on debug mode. Any errors encountered are logged to the log file. If you specify this option, you can use Ctrl+c to shutdown Run-Time Manager. |
| -a <i>flapp_dir</i> | Defines the full path of the directory containing the application files. This path overrides any path set by the FLAPP environment variable. |
| -p <i>flink_dir</i> | Defines the full path of the directory containing the FactoryLink programs. This path overrides any path set by the FLINK environment variable. |
| -f1 | PID check. If experiencing kernel lock-up problems, the switch adds extra checking to prevent rogue tasks from corrupting the kernel; however, there is a performance penalty. |
| -l <i>log_file</i> | Logs error and other data to <i>log_file</i> . |
| -t <i>timeout</i> | Defines the start/stop timeout, in seconds, for the Run-Time Manager error report process. The default timeout is 60 seconds |
| -s | Starts only the shared domain on a PC platform. The user domain is not started. |
| -v | Turns on verbose mode for .CT generation. |
| -n <i>fldomain</i> | Defines the domain name, where <i>domain</i> can either be shared or user. If you specify shared, only the shared domain is started. This overrides the FLDOMAIN environment variable. |
| -i <i>fname</i> | Defines the name of the application to start. This overrides the FLNAME environment variable. |

- **USING RUN-TIME MANAGER**
- *Setting Command Line Options*
-
-

| Option | Action |
|----------------------|---|
| -u <i>fluser</i> | Defines the user name. This overrides the FLUSER environment variable. |
| -w <i>warm_start</i> | Turns on the warm start mode to reload persistent values. If you specify this option, FactoryLink loads persistent tags with the last value saved for them. |

The options you specify using the procedures in this chapter apply to all tasks started for the application. You can set options for individual tasks that override these defaults using the System Configuration module which is accessible from the Configuration Manager Main Menu.

Where you define the default run-time options depends on whether you are starting the Run-Time Manager from a FactoryLink icon or from an operating system command line.

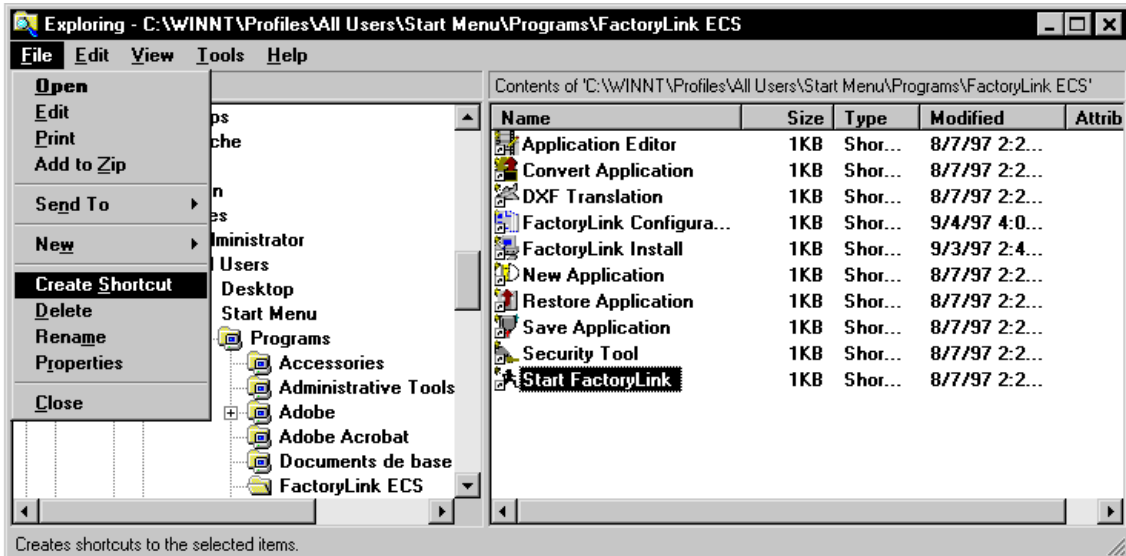
If you are starting Run-Time Manager from a command line, use the following syntax.

`flrun options`

where *options* is one or more of the options specified in the previous table. Each option is separated by a space. Refer to “Accessing Run-Time Manager” on page 299 for platform-specific details on starting Run-Time Manager from the command line.

If you are starting Run-Time Manager from a Windows icon, you must create a shortcut icon for the Run-Time Manager icon using the following procedure.

- 1 Choose drivename:>Winnt>Profiles>All Users>Start Menu>Programs>FactoryLink 6.5.0 from the Explorer dialog.
- 2 Click on Start FactoryLink in the Name list to highlight.
- 3 Choose Create Shortcut from the File menu.



A copy of FactoryLink 6.5.0 displays in the Name list.

- 4 Drag this copy to the Desktop Manager.
- 5 Right-click on the new shortcut icon and select Properties from the pop-up menu.

- **USING RUN-TIME MANAGER**
- *Setting Command Line Options*
-
-

6 Select the Shortcut tab to display the property sheet.



7 Add the desired options at the end of the Target field.

8 Click OK to close the property sheet.

A Start FactoryLink icon displays on the Desktop Manager. You can now access the Run-Time Manager by double clicking on this icon.

ACCESSING RUN-TIME MANAGER

How you open the Run-Time Manager depends on your operating system platform. Once opened, Run-Time Manager acts the same for all platforms. The method for starting Run-Time Manager for each platform follows.

On Windows NT and Windows 95 Platforms

- 1 Click on the FactoryLink icon from the Desktop Manager to display the FactoryLink 6.5.0 program group.



Double click here
to start
the Run-Time
Manager

If you have more than one FactoryLink application, be sure to open the program group for the desired application.

- 2 Click on the Start FactoryLink icon. This starts both the shared and user domains.
- Or

Double click on the Start FactoryLink icon on the Desktop Manager.

On OS/2 Platform

If you are running FactoryLink on an OS/2 Windows operating system, you can open the Run-Time Manager from either the Desktop Manager or the command line. Both methods start the shared and user domains. The two methods are described below.

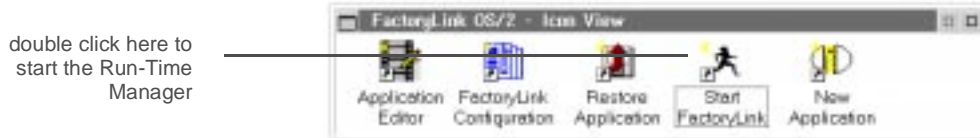
- **USING RUN-TIME MANAGER**

- *Accessing Run-Time Manager*

-
-

From Desktop Manager

- 1 Choose FactoryLink OS/2 from the Desktop Manager menu to display the applications available with the FactoryLink group.



- 2 Double click the left button to open Run-Time Manager.

From a Command Line

Enter the following command at the system prompt to open Run-Time Manager from an OS/2 command line.

`flrun options`

where *options* is one or more of the options specified in “Setting Up Program Arguments” on page 293.

On UNIX Platform

Perform the following procedure to open the Run-Time Manager if you are running FactoryLink on a UNIX platform.

- 1 Open two operating system windows, one to run FactoryLink in the shared domain and another to run it in the user domain. You must open the shared domain before you open the user domain.
- 2 Ensure the FLAPP, FLINK, FLDOMAIN, FLNAME, AND FLUSER environment variables are set for the shared domain. If not, set them. How you do this depends on whether you are using a C shell or a Korn or Bourne shell.

If you are using a C shell, enter the following commands at the system prompt of the shell where you are starting the shared domain.

```
setenv FLINK flink_dir
setenv FLAPP flapp_dir
setenv FLDOMAIN domain
setenv FLNAME app_name
setenv FLUSER user_name
```

If you are using a Korn or Bourne shell, enter the following commands at the system prompt of the shell where you are starting the shared domain.

```
FLINK=flink_dir
FLAPP=flapp_dir
export FLAPP
FLDOMAIN=domain
export FLDOMAIN
FLNAME=app_name
export FLNAME
FLUSER=user_name
export FLUSER
```

where

- flink_dir* defines the full path to the directory containing the FactoryLink program files.
- flapp_dir* defines the full path to the directory containing your application files.
- domain* defines the domain you are starting. This should be shared for the window where you are starting the shared domain.
- app_name* defines the full path to the directory containing your application files.
- user_name* defines the log on name of the user starting the application.

3 Enter the following command at the system prompt.

```
flrun options
```

where *options* is one or more of the options specified in the table on page 293. The system starts the Run-Time Manager in the shared domain, but the Run-Time Manager screen is not yet visible.

- **USING RUN-TIME MANAGER**

- *Accessing Run-Time Manager*

-
-

- 4 Ensure the FLAPP, FLINK, FLDOMAIN, FLNAME, AND FLUSER environment variables are set for the user domain. If not, set them. How you do this depends on whether you are using a C shell or a Korn or Bourne shell.

If you are using a C shell, enter the following commands at the system prompt of the shell where you are starting the user domain.

```
setenv FLINK flink_dir
setenv FLAPP flapp_dir
setenv FLDOMAIN domain
setenv FLNAME app_name
setenv FLUSER user_name
```

If you are using a Korn or Bourne shell, enter the following commands at the system prompt of the shell where you are starting the user domain.

```
FLINK=flink_dir
FLAPP=flapp_dir
export FLAPP
FLDOMAIN=domain
export FLDOMAIN
FLNAME=app_name
export FLNAME
FLUSER=user_name
export FLUSER
```

where

| | |
|------------------|---|
| <i>flink_dir</i> | defines the full path to the directory containing the FactoryLink program files. |
| <i>flapp_dir</i> | defines the full path to the directory containing your application files. |
| <i>domain</i> | defines the domain you are starting. This should be user for the window where you are starting the user domain. |
| <i>app_name</i> | defines the full path to the directory containing your application files. |
| <i>user_name</i> | defines the log on name of the user starting the application. |

- 5 Enter the following command at the system prompt:

`flrun options`

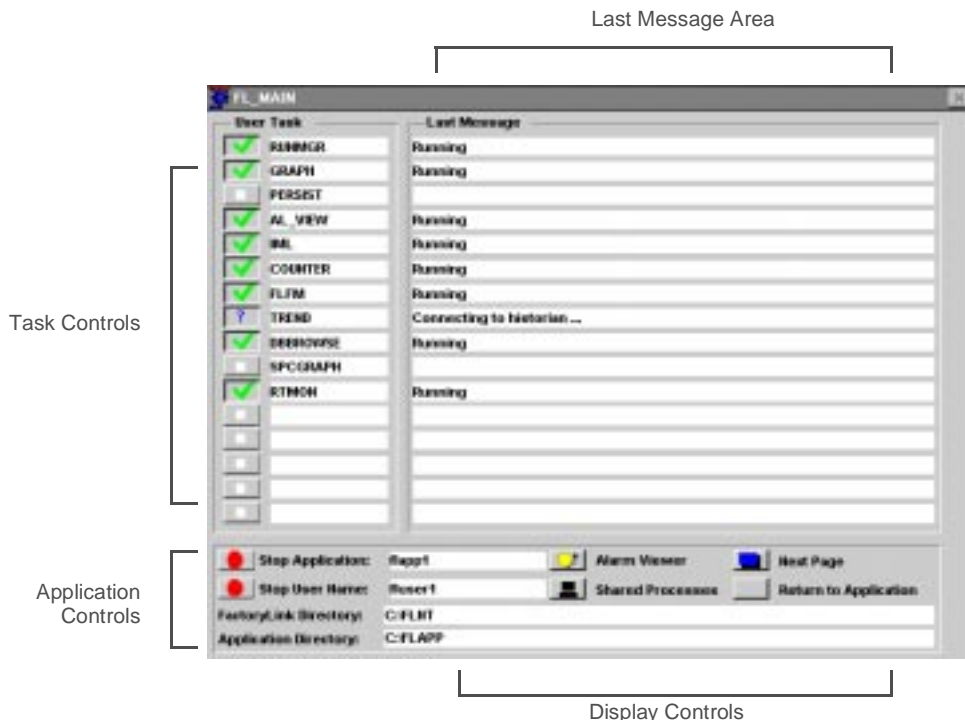
where *options* is one or more of the options specified in the table on page 293.

The system starts the Run-Time Manager in the user domain and the Run-Time Manager screen opens. You can toggle between the Run-Time Manager for the shared domain and that for the user domain from the Run-Time Manager screen. Refer to the “Run-Time Manager User-Interface Screen” on page 304 for more details.

- USING RUN-TIME MANAGER
- Run-Time Manager User-Interface Screen
-
-

RUN-TIME MANAGER USER-INTERFACE SCREEN

When you start the Run-Time Manager, the Run-Time Manager main screen is displayed. This document displays the Run-Time Manager screen as it is shipped with FactoryLink. You can customize the Run-Time Manager screen according to your needs using the Application Editor.

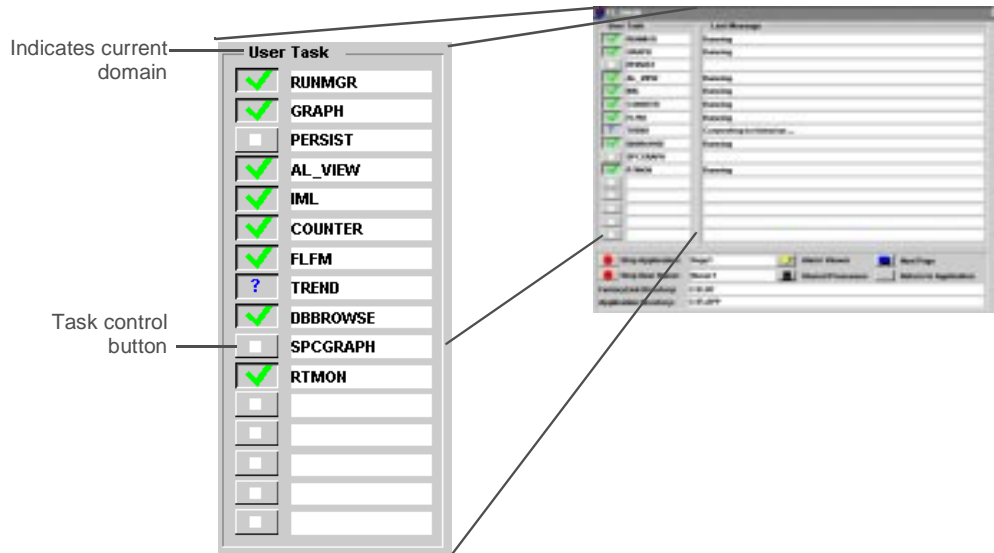


The screen has four major components.

- **Task controls**—Turns tasks on or off. Refer to the “Task Controls” on page 305 for more details.
- **Last message area**—Shows the last message sent by the task. Refer to the discussion on the task in the *FactoryLink Configuration Guide* for more details.
- **Application controls**—Displays information about the current application and turns the application off. Refer to the “Application Controls” on page 306 for more details.
- **Display controls**—Provides access to other application screens. Refer to the “Run-Time Manager User-Interface Screen” on page 304 for more details.

Task Controls

The task control area lists the available tasks for your application. Up to 31 tasks are available in both the shared and user domain. If you are in the shared domain, this area is titled Shared Task. If you are in the user domain, this area is titled User Task.



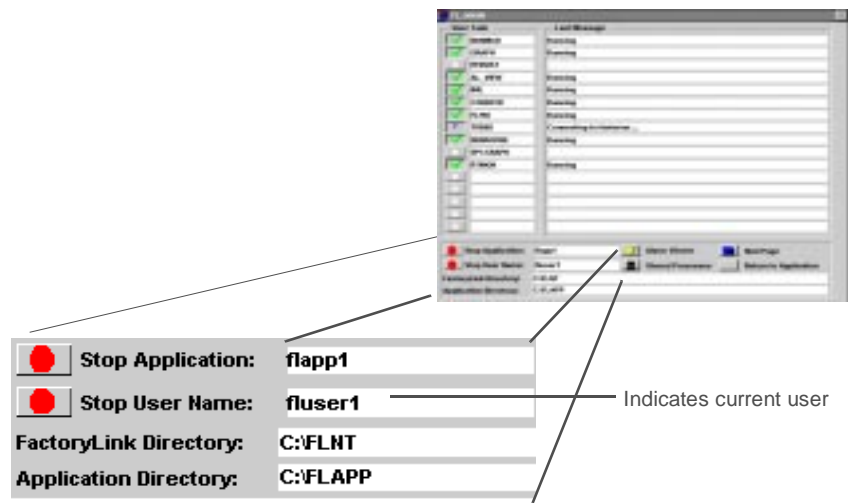
The contents of the button to the left of the task name indicates the status of the task. This can be one of the following.

- Empty Task is not running. If you choose this button when the task is not running, the task starts.
- Green check mark Task is running. If you choose this button when the task is running, the task stops.
- Yellow question mark Task is stopping.
- Blue question mark Task is starting.
- Red X Task has reported an error.

- **USING RUN-TIME MANAGER**
- *Run-Time Manager User-Interface Screen*
-
-

Application Controls

The application control area provides information about the application running and provides the ability to stop the application or the user instance.



- Stop Application

Stops the application. Always choose this button if you are running in a single-user environment. If you running in a multi-user environment, this stops the application for all users.
- Stop User Name

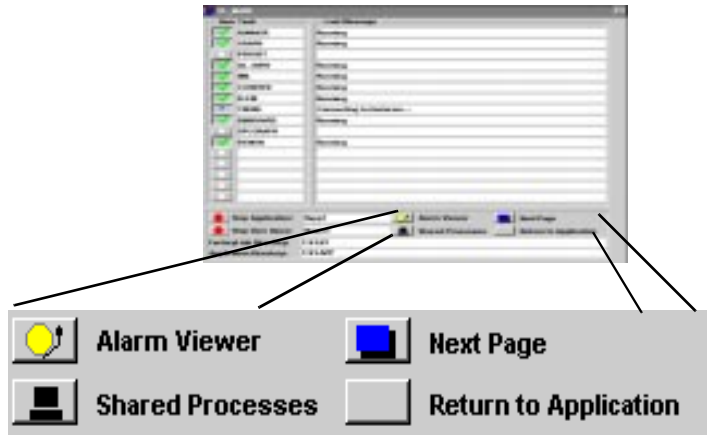
Stops the application for the current user. Always choose this button if you are running in a multi-user environment. If this is the shared domain, shareusr is displayed in this field. If this is the user domain, fluser is displayed in this field indicating the instance of the user domain. In a single-user environment, this will always be fluser1.
- FactoryLink Directory

Displays the current FactoryLink directory.
- Application Directory

Displays the current application directory.

Display Controls

The display control area provides the ability to open other user-interface screens for the application.



- | | |
|-----------------------|--|
| Alarm Supervisor | Choose this button to open the alarm summary display. |
| Next Page | Choose this button to display a second page of tasks. This is necessary only if all the tasks in your application do not fit on a single display. |
| Shared Processes | <p>Choose this button to toggle between the user domain and the shared domain.</p> <p>If you are in the user domain, this button is titled Shared Processes. If you choose this button, you open the shared domain.</p> <p>If you are in the shared domain, this button is titled User Processes. If you choose this button, you open the user domain.</p> |
| Return to Application | <p>Choose this button to return to the application interface screen.</p> <p>You must animate this button to do this when you are configuring your application.</p> |

Note: If you are running a demonstration system, the words DEMO SYSTEM appear in the title bar of the Run-Time Manager console window. A demo system shuts down after running for one hour. To view the licensing options for your FactoryLink system, use the ukey utility. The demo system is option 89 (refer to “UKEY” on page 411 for more information about ukey).

- **USING RUN-TIME MANAGER**

- *Run-time Manager Messages*

-
-

RUN-TIME MANAGER MESSAGES

Error messages are displayed on the screen if errors occur while using Run-time Manager. This section lists these messages, describes their cause, and provides suggested actions.

Messages are described in two ways:

- Text message—Refer to the “Text Messages” on page 308 for text messages.
- Error numbers—Refer to the “Messages with Error Numbers” on page 314 for error number messages.

Text Messages

Bad command *number*

Cause: An invalid command was written to the global element COMMAND.

Action: Verify the external process' compatibility with FactoryLink.

Bad file size for *filename*

Cause: The FLAPP/CT/TYPE.CT file has been damaged.

Action: Delete the file and re-start the application to rebuild the file.

Bad index in file *filename*

Cause: Unable to read the index of the FLAPP/CT/TYPE.CT file.

Action: Delete the file and re-start the application to rebuild the file.

Can't convert subgroup table *name*

Cause: Tried to convert an older SPC application.

Action: See the conversions instructions in the Release Notes.

Can't create appl. *filename* error *error number*

Cause: Tried to start an application already started.

Action: No action required.

Can't open file *filename*

Cause: The disk may be full, or this may be a log file opened by another process.

Action: Delete unnecessary files. If this error occurs often, additional disk space may be required.

Can't read options file *FL.OPT*

Cause: FL.OPT has been damaged.

Action: Verify the contents of FL.OPT by running UKEY. Copy flnew.opt to fl.opt.

Can't start process *process name*

Cause: The binary file may not exist, may not be executable, or the filename in the EXECUTABLE FILE field in the System Configuration Table may be incorrect. Under some system platforms this may indicate an insufficient number of processes.

Action: Verify the correct file name in the System Configuration Table or see the *Fundamentals* manual.

Can't stop process *process name*

Cause: The process may already have stopped.

Action: Attempt to stop the process manually if it has not stopped.

Client *processes* failed to start

Cause: One or more processes could not be started.

Action: See the error messages for the particular process.

Client *processes* failed to stop

Cause: The process may already have stopped.

Action: Attempt to stop the process manually if it has not stopped.

Directory *directory name* does not exist

Cause: Can't find FLAPP directory specified by the environment variable.

Action: Set FLAPP to a valid application directory.

- **USING RUN-TIME MANAGER**

- *Run-time Manager Messages*

-
-

Directory *directory name* is not a valid FactoryLink directory

Cause: FLAPP does not have a valid subdirectory structure.

Action: (1) Check the FLAPP directory. Add the missing subdirectories.
(2) Set FLAPP to a valid application directory.

Domain *domain name* can only have one instance

Cause: A “parent” domain has been configured as having more than one instance.

Action: A “parent” domain may only have one instance. Open the Domain Element List and change the entry in the #INST field to 1.

Domain *domain name* isn’t in the domain CT

Cause: The specified domain does not exist in the domain .CT.

Action: The domain name may have been entered incorrectly or may not exist in the Domain Element List. Verify the domain exists in the Domain Element List and has been entered correctly.

Environment Tag *element name* has an invalid type

Cause: The wrong data type was specified for the field.

Action: Enter the correct data type for this field.

FactoryLink initialization failed

Cause: FactoryLink system failed to initialize.

Action: Ensure the master key is present and properly connected. Run KEYINST and FLKEYVAL.

FactoryLink system monitoring failed

Cause: The system was unable to start the monitor task. The system may not contain sufficient memory.

Action: Stop unnecessary processes.

Grace period has expired. Software must be registered

- Cause: FactoryLink was installed without registering it within the 10 day grace period.
- Action: Run FLKEYVAL and follow the instructions for registration.

Kernel initialization failed

- Cause: FactoryLink was already initialized. FactoryLink was already running when start-up was attempted.
- Action: No action required.

Monitor tag array definition failed

- Cause: A System Configuration Table monitor element (Start Trigger, Task Status, or Task Message) may be undefined or defined incorrectly for one of the tasks being started.
- Action: Open the System Configuration Table and define the elements correctly.

Number of defined proc *process number* more than max *maximum number*

- Cause: More than 31 processes were started.
- Action: Start fewer processes.

Out of RAM

- Cause: No more memory is available.
- Action: Close any unnecessary windows or programs. Add more memory to the system if this error occurs often.

Output tag array definition failed

- Cause: A System Configuration Table output element (Display Status, Display Name, or Display Description) may be undefined or defined incorrectly for one of the tasks being started.
- Action: Open the System Configuration Table and define the elements correctly.

- **USING RUN-TIME MANAGER**

- *Run-time Manager Messages*

-
-

Process *process name* may not have started

Cause: The process failed to become active. It may not have been able to register with FactoryLink.

Action: Verify the filename in the EXECUTABLE FILE field in the System Configuration Table. Verify the process' compatibility with FactoryLink.

Process *process name* may not have stopped

Cause: The process may already have stopped.

Action: Attempt to stop the process manually if it has not stopped.

Read failed on file *filename*

Cause: A read operation on the named file failed.

Action: Verify configuration table entries, communication parameters, hardware identification information, and electrical connections.

Read header failed on file *filename*

Cause: The .CT file may be damaged.

Action: Delete the .CT file in the FLINK/ct directory and re-start the application to rebuild the file.

Real-time database isn't initialized

Cause: FactoryLink is starting or shutting down.

Action: Start FactoryLink again.

Real-time database for *application* doesn't exist

Cause: An attempt was made to start a USER domain when the SHARED domain did not exist.

Action: Start the SHARED domain.

Run-time Manager: *errno* = *error number*

Cause: The Run-time Manager was awakened when there was nothing for it to do. (Internal error.)

Action: No action required.

Run-time Manager CT processing failed

Cause: One or more of the elements ARGUMENT, COMMAND, PASSWORD, SHUTDOWN, or STARTUP is not defined in the GLOBAL.CDB file. The file has been damaged or installed incorrectly.

Action: Contact Customer Support.

Run-time Manager failed to start

Cause: The Run-time Manager must be the first task started. The Start Order number for RUNMGR must be 0; no other task may have a zero for a Start Order number.

Action: Check the System Configuration Table to verify the Start Order.

Run-time Manager failed to stop

Cause: The process may already have stopped.

Action: Attempt to stop the process manually if it has not stopped.

Run-time Manager is already running

Cause: A copy of the Run-time Manager is already running for the DOMAIN and USER name specified.

Action: Change the DOMAIN or USER name.

Software has not been enabled

Cause: An error has occurred with the fl.key file in the opt directory.

Action: 1. Run KEYINST and if an invalid or archaic agreement is found, re-enter the configuration information.
2. Run FLKEYVAL and follow the instructions for registration.

Write error on log file *filename*

Cause: The disk may be full.

Action: Delete unnecessary files.

- **USING RUN-TIME MANAGER**

- *Run-time Manager Messages*

-
-

Messages with Error Numbers

Two Run-time Manager error messages display error numbers within the message:

Can't create appl. *filename* error *error number*
Run-time Manager: *errno* = *error number*

The error numbers in these messages are generated by the FactoryLink real-time database. Each error number has a corresponding error keyword. Even though this keyword is not displayed in the error message, knowing the error keyword helps Customer Support engineers identify the cause of the problem.

1 FLE_INTERNAL

Cause: Internal Error.

Action: See “Correcting Internal Errors” at the end of this section.

2 FLE_OUT_OF_MEMORY

Cause: There is not enough RAM available.

Action: Add RAM to the system.

4 FLE_NO_FLINK_INIT

Cause: The real-time database has not been created or cannot finish initializing.

Action: Shut down FactoryLink and re-start it.

5 FLE_NO_PROC_INIT

Cause: One of the following conditions caused this error:
The task name of the specified task has not been entered in the System Configuration Table.

Action: Perform the corresponding action:
Enter the task name in the Task Name field of the System Configuration Table.

Cause: The task is already running.

Action: Nothing—the task will continue to run.

Cause: The indicated task is not enabled on the FactoryLink key.

Action: Contact your FactoryLink sales representative to obtain the proper key and/or option.

7 FLE_BAD_ARGUMENT

Cause: Internal error—a task tried to pass an invalid argument to the real-time database.

Action: See “Correcting Internal Errors” at the end of this section.

9 FLE_BAD_TAG

Cause: The .CTs need to be rebuilt.

Action: Rebuild the .CTs by running CTGEN.

10 FLE_NULL_POINTER

Cause: Internal error—a task gave out a null pointer.

Action: See “Correcting Internal Errors” at the end of this section.

12 FLE_PROC_TABLE_FULL

Cause: 31 tasks are running in the chosen domain. Only 31 tasks can run in each domain at once.

Action: No action required—do not try to run more than 31 tasks per domain at a time.

13 FLE_BAD_PROC_NAME

Cause: One of the following conditions caused this error:
The .CTs need to be rebuilt.

Action: Perform the corresponding action:
Run CTGEN to rebuild the .CTs. If this does not solve the problem, proceed to step 2.

Cause: A task name used in the application has not been entered in the Task Name field of the System Configuration Table.

Action: Open the System Configuration Table from the Configuration Manager Main Menu and ensure the names of all tasks in the application are present in the Task Name field.

- **USING RUN-TIME MANAGER**

- *Run-time Manager Messages*

-
-

Cause: The requested task is not running.

Action: Ensure you started all tasks needed to run the application.

14 FLE_BAD_USER_NAME

Cause: The FLUSER environment variable is not set.

Action: Set the environment variables; re-start FactoryLink.

22 FLE_ALREADY_ACTIVE

Cause: The task is already running.

Action: None-do not start a task already running.

23 FLE_NOT_LOCKED

Cause: Internal error—a task tried to unlock the real-time database without having locked it first.

Action: See “Correcting Internal Errors” at the end of this section.

24 FLE_LOCK_FAILED

Cause: Internal error—a task used an invalid Task ID to lock the real-time database. Therefore, the task did not successfully lock the real-time database or perform its function.

Action: See “Correcting Internal Errors” at the end of this section.

25 FLE_LOCK_EXPIRED

Cause: A task has kept the real-time database locked longer than the kernel allows.

Action: None—currently, the kernel allows tasks to lock the real-time database for as long as necessary. Therefore, the lock time will not expire.

26 FLE_WAIT_FAILED

Cause: Internal error—while trying to do a change-wait on an element, a task sent an invalid Task ID to the real-time database.

Action: See “Correcting Internal Errors” at the end of this section.

28 FLE_QSIZE_TOOBIG

Cause: Internal error—a task attempted to attach a queue to an element, but there was not enough memory.

Action: See “Correcting Internal Errors” at the end of this section.

29 FLE_QSIZE_CHANGED

Cause: Internal error—a task attempted to attach a queue to an element, but a queue of a different size was already attached.

Action: See “Correcting Internal Errors” at the end of this section.

30 FLE_NO_TAG_LIST

Cause: This error only occurs on custom tasks written using the FactoryLink PAK. If the custom task was set up to access elements by name instead of by ID number, this error can occur if the element list has not been defined.

Action: Use the API function FL_SET_TAG_LIST to define the element list. Then, re-start FactoryLink. Refer to the *Programmer's Access Kit (PAK)* manual for information about L_SET_TAG_LIST.

31 FLE_TAG_LIST_CHANGED

Cause: This message can be displayed only if an application has been set up so one task monitors another task's element list. If one task modifies the other task's element list, the task that modified the list will return this message, thus informing the task that its list has been modified.

Action: No action required.

32 FLE_WAKEUP_FAILED

Cause: Internal error—The real-time database tried unsuccessfully to wake a task waiting for the value of a particular element to change.

Action: See “Correcting Internal Errors” at the end of this section.

- **USING RUN-TIME MANAGER**

- *Run-time Manager Messages*

-
-

33 FLE_NO_SIGNALS

Cause: If a task calls the FL_RECV_SIG function to find out if any signals have been sent to it, the FL_RECV_SIG function returns a message. If no signals have been sent to the task, FL_RECV_SIG returns this message.

Action: No action required.

34 FLE_SIGNALLED

Cause: The real-time database sends this message to inform a sleeping task that the task has received a signal, not a change in value of the element for which it is waiting.

Action: No action required.

35 FLE_NOT_MAILBOX

Cause: One of the following conditions caused this error:
The .CTs need to be rebuilt.

Action: Perform the corresponding action:
Run CTGEN to rebuild the .CTs. If this does not solve the problem, proceed to step 2.

Cause: While configuring a task, you entered an element of a data type other than MAILBOX in a field that requires a MAILBOX element.

Action: Open the configuration panels from the Main Menu of the task containing the error. Ensure the data types of the elements in all fields requiring MAILBOX elements, are defined as MAILBOX.

36 FLE_NO_MESSAGES

Cause: If through the PAK_QUERY_MBX function, a task requests to view a message whose queue number falls outside the range of available messages, the QUERY_MBX function returns this message. For example, if a task requests to view the third message waiting in the queue, and only two messages are in the queue, then the task is requesting to view a message outside the range of available messages.

Action: No action required.

37 FLE_ACCESS_DENIED

Cause: If a task tries to read a mailbox element that it is not allowed to read, the real-time database returns this error. If the task is a custom task developed using PAK, the programmer may not have set up ownership of the mailbox element by the task requesting to read from it.

Action: Use the PAK function FL_SET_OWNER_MBX to establish the task as owner of the mailbox element.

Cause: Internal error.

Action: See “Correcting Internal Errors,” at the end of this section.

41 FLE_APP_EXISTS

Cause: A real-time database with the same FLNAME but a different number of users or elements already exists.

Action: Either shut down the running real-time database or set FLNAME with a different name.

42 FLE_NO_FLINK_RTDB

Cause: A user domain was started for a real-time database for which the shared domain has not been started.

Action: Start the shared domain first.

- **USING RUN-TIME MANAGER**

- *Run-time Manager Messages*

-
-

Correcting Internal Errors

Internal errors are generally caused by one of the tasks in the system in use and not by your application. For this reason, we recommend you use the following guidelines to correct an internal error:

1. Try to determine which task is sending the error by shutting down FactoryLink, restarting it, and starting each task, one at a time.
2. Write down any error messages displayed on the Run-time Manager screen and their corresponding tasks. (The task having the problem may generate a seemingly unrelated error message.)
3. Contact the supplier of the task in error if the task in error is an external task.
4. Contact Customer Support if the task in error is a FactoryLink task.
5. Contact Customer Support if the task cannot be identified.

Using Run-Time Monitor

Run-Time Monitor (RTMON) allows you to monitor how the real-time application is functioning in order to test your application before using it or to debug problems that arise during processing after it is developed. Using RTMON, you can:

- Simulate input to real-time database elements.
- Monitor how the values of data elements in the real-time database change.
- Start, stop, or monitor FactoryLink tasks.

In general, any development, testing, or debugging operation that can be aided by directly accessing the FactoryLink real-time database, either by reading from or writing to elements, is made easier with RTMON.

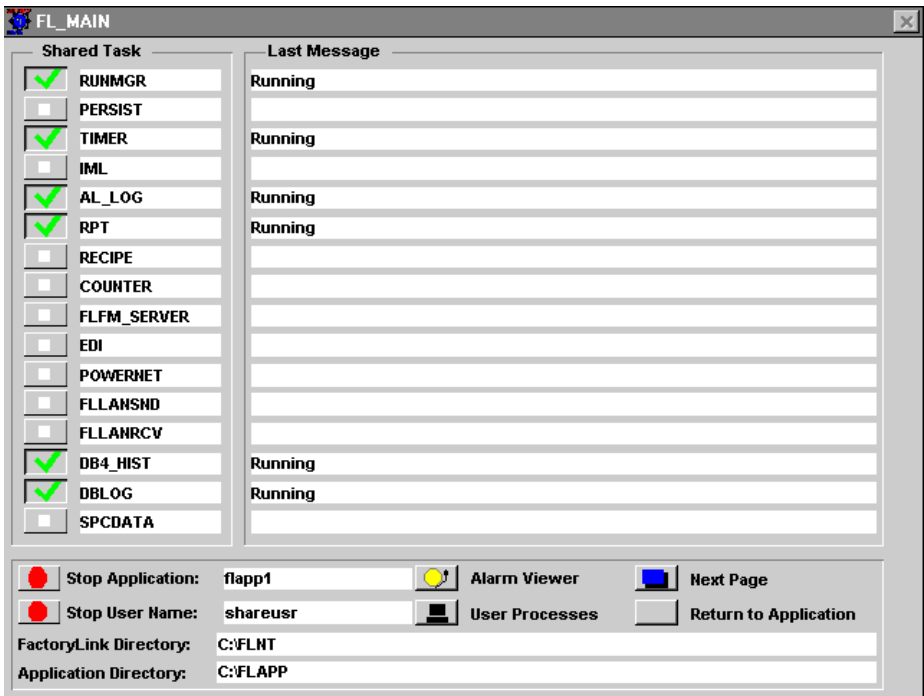
While developing an application, periodically run and monitor it using RTMON to ensure the application is functioning as you intended.

- **USING RUN-TIME MONITOR**
- *Accessing the Run-Time Monitor*
-
-

ACCESSING THE RUN-TIME MONITOR

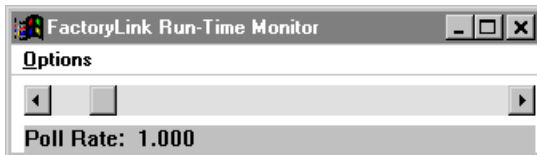
Perform the following steps to open RTMON.

- 1 Start up the FactoryLink application. How you do this depends on your operating system platform. Refer to “Using Run-Time Manager” on page 293 for more details on this operation. The following screen is displayed when you start the application.



- 2 Choose User Processes from the Display Control area if you want to monitor the user domain. The best domain in which to run RTMON depends on the type of information you are monitoring. Use the following guidelines to decide on the best domain.
 - If you are monitoring elements, run RTMON in the user domain. This enables you to monitor elements from both the shared and user domains.
 - If you are monitoring FactoryLink tasks, run RTMON in the domain in which the desired task(s) runs. For example, to monitor the status of the Alarm Supervisor task, run RTMON in the shared domain.

- 3 You may have to choose Next Page from the Display Control area to see the RTMON task in the Task Control list.
- 4 Choose the button next to RTMON in the Task Control list. The Run-Time Monitor panel opens.



This panel consists of two parts:

- **Menu Bar**—Provides access to a group of tools that permit you to use RTMON through the Options menu. The following pulldown menu is displayed when you choose Options from the menu bar..



Each item on this pulldown menu is discussed in the following pages.

- **Poll Rate Bar**—The Poll Rate bar is a scroll bar that allows you to indicate the rate at which RTMON polls the processes it is monitoring. The Poll Rate bar can specify from .10 (one tenth) of a second to 10 seconds. Move the slider bar located on the scroll bar to change the poll rate.

The remainder of this chapter describes how to use the Run-Time Monitor.

- **USING RUN-TIME MONITOR**
- *Viewing the Current Value of Elements*
-
-

VIEWING THE CURRENT VALUE OF ELEMENTS

You can view the current value of elements using the watch list display. This display is real-time and updates at the selected poll rate.

- 1 Choose Watch from the Options pulldown menu to open the watch list . The Monitor Watch List panel is displayed.



You can display multiple watch lists at the same time. Each time you choose Watch from the Options pulldown menu, a new watch list panel is created. The watch list panel has one item on its menu bar.

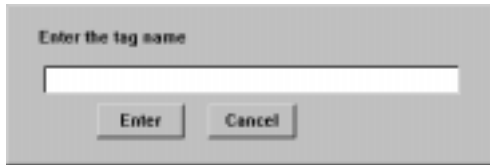
- 2 Choose Options from the watch list menu bar to display the following pulldown menu.



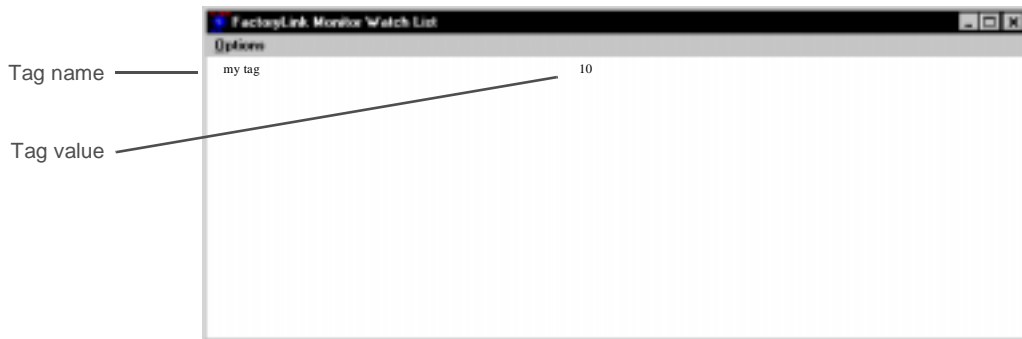
The actions you can perform on the watch list are described in the following pages. The order they are discussed is the order you would use them to build and save a watch list rather than the order they are presented in the pulldown menu.

Adding Elements to a Watch List

- 1 Choose Add Watch from the watch list Options pulldown menu. The following dialog is displayed.



- 2 Enter the name of the tag you want to monitor and choose the Enter button. Its name and current value are added to the watch list display.



- 3 Repeat step 2 for each element you want to monitor. If you list an element name more than once or on more than one watch list panel, only the first entry is updated.
- 4 Choose Cancel from the dialog to close it when you are finished adding elements to the watch list.

- **USING RUN-TIME MONITOR**
- *Viewing the Current Value of Elements*
-
-

Deleting Elements from a Watch List

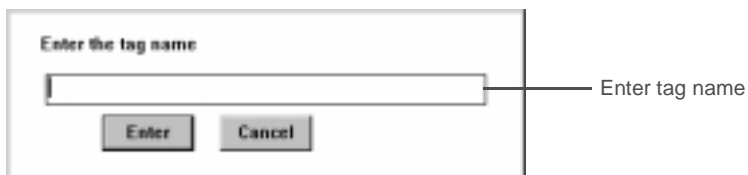
- 1 Choose Delete Watch from the watch list Options pulldown menu. The following dialog is displayed.



- 2 Enter the name of the tag you want to delete and click on Enter. Its name and current value are deleted from the watch list display.
- 3 Choose Cancel from the dialog to close it when you are finished deleting elements from the watch list.

Finding Elements in a Watch List

- 1 Choose Find Watch from the watch list Options pulldown menu. The following dialog is displayed.



- 2 Enter the name of the tag you want to find and choose the Enter button. The first occurrence of the element is found in the watch list.
- 3 Choose Cancel from the dialog to close it after you have finished finding elements in the watch list.

Storing Watch List

The elements added to a watch list remain in the list for as long as the Run-Time Manager is active. Once you close the Run-Time Manager, the watch list is discarded. To prevent the necessity of rebuilding a watch list each time you want to monitor a set of data elements, store the contents of the watch list to a file before exiting the Run-Time Manager. Perform the following steps to store the contents of a watch list.

- 1 Add the elements you want included in the watch list to the watch list display. Refer to “Adding Elements to a Watch List” on page 325 for details on how to do this.
- 2 Choose one of the following options from the watch list Options pulldown menu.

| | |
|-----------|---|
| Save TAGS | Saves the tag names of elements defined in the watch list. Use this option if you want to see the values of these tags as they currently exist in the real-time database when you restore this list. |
| Save Data | Saves the tag names of elements defined in the watch list and their current values. Use this option if, when you restore this list, you want to restore the values of these tags as they exist when the save is made. |

If you choose Save TAGS, the following dialog is displayed requesting the name of the file to receive the watch list elements. If you choose Save Data, a similar dialog is displayed requesting the data filename.



- 3 Enter the name of the file to receive the tags.
- 4 Choose Cancel from the dialog to close it when you have completed saving all watch lists.

- **USING RUN-TIME MONITOR**
- *Viewing the Current Value of Elements*
-
-

Retrieving Watch List

Perform the following steps to retrieve the elements in a watch list saved with the Save TAGS or Save Data option.

- 1 Choose one of the following options from the watch list Options pulldown menu.
 - Load TAGS Retrieves the tag names of elements saved in a file using the Save TAGS option and writes them to the watch list. The values of these elements display as they currently exist in the real-time database.
 - Load Data Retrieves the tag names of elements saved in a file using the Save Data option and writes them to the watch list. The values of these elements display as they existed in the real-time database when the save was made.

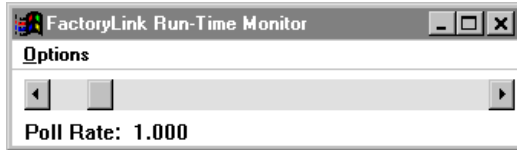
If you choose Load TAGS, the following dialog is displayed requesting the name of the file containing the watch list elements. If you choose Load Data, a similar dialog is displayed requesting the data filename.



- 2 Enter the name of the file containing the tags you want to retrieve.
- 3 Choose Cancel from the dialog to close it when you have completed loading all watch lists.

READING AND WRITING A REAL-TIME DATABASE ELEMENT

- 1 Choose Options from the Run-Time Monitor panel.



The following pulldown menu is displayed.



- 2 Choose Tag Input from the Options pulldown menu. The Tag Input dialog is displayed.



You can display multiple tag input panels. Each time you choose Tag Input, a new panel is displayed.

- 3 Provide the following information.

Tag Name Enter the logical name assigned to the real-time database element you want to read or write. When you enter a name, the location of the element in the real-time database referenced by the tag is displayed in the Tag Number field.

Value Enter the value you want written to the element if you want to write a value to the element.

- 4 Choose the command button that represents the operation you want to perform. This can be one of the following.

- **USING RUN-TIME MONITOR**
- *Reading and Writing a Real-Time Database Element*
-
-

Read Reads the element. The current value of the element is displayed to the right of the Value field.



Write Writes the value in the Value field to the element.

Force Force writes the value in the Value field to the element.

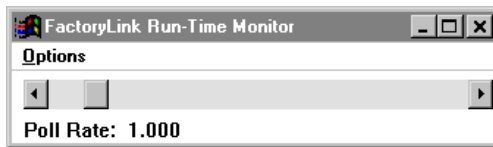
MONITORING FACTORYLINK PROCESSES

You can use RTMON to monitor the activity of all FactoryLink tasks in an application. This is useful for detecting and diagnosing problems during application development. For example, if you detect most FactoryLink tasks have little or no status changes except Math & Logic, this could indicate one of the following:

- At least one Math & Logic procedure is using up all CPU resources.
- Math & Logic is caught in a loop. This can happen if a procedure tests for condition that never occurs.

Perform the following steps to monitor FactoryLink tasks.

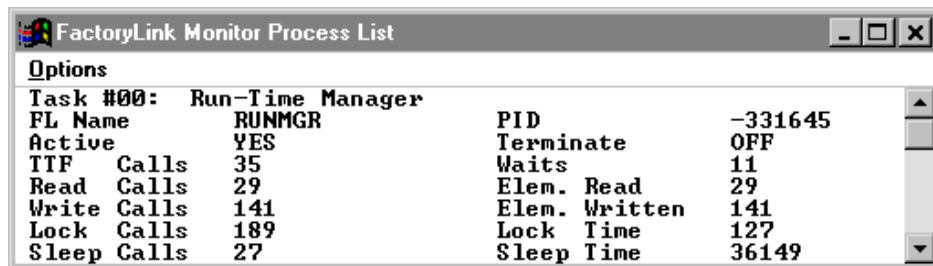
- 1 Choose Options from the Run-Time Monitor panel.



The following pulldown menu is displayed.



- 2 Choose Process from the Options pulldown menu. The Monitor Process List window is displayed.



- **USING RUN-TIME MONITOR**
- *Monitoring FactoryLink Processes*
-
-

The following information is provided about each FactoryLink task.

| | |
|---------------|---|
| Task # | Number and descriptive name of a FactoryLink task. |
| FL Name | Short name used to reference the FactoryLink task. |
| Active | Indicator of whether or not the specified task is active. |
| TTF Calls | Number of times the task has checked its terminate flag since startup. The terminate flag is a bit in the kernel. The task reads the value of its terminate flag periodically to determine whether it should shut down or keep running. If the terminate flag is set to 1 (ON), the FactoryLink task sets its start trigger to 0 (OFF). When the start trigger is set to 0 (OFF), the Run-Time Manager shuts the task down. Each task has its own terminate flag and start trigger. |
| Read Calls | Number of read operations the task has performed since startup. |
| Write Calls | Number of write operations the task has performed since startup. |
| Lock Calls | Number of times the task has locked the real-time database since startup. When a task reads from or writes to the real-time database, it locks all other tasks out of the database. When the task has completed its operations, it unlocks the real-time database so another task can open it. |
| Sleep Calls | Number of times the task has slept while waiting for elements to change since startup. |
| PID | Process ID number (usually decimal) assigned to the task by the operating system. |
| Terminate | Indicator of whether or not the task's terminate flag is set. |
| Waits | Number of times the task has waited for access to the real-time database since startup. |
| Elem. Read | Number of elements read by the task since startup. |
| Elem. Written | Number of elements the task has written since startup. |
| Lock Time | Number of milliseconds the task has had the real-time database locked. |
| Sleep Time | Total number of milliseconds the task has been asleep since task startup. This value is updated after each sleep cycle. |

This panel has one item on its menu bar. Choose Options from the process list menu bar. The following pulldown menu is displayed.



When you choose one of these options, it acts on the task currently displayed in the process list window. These are:

- Start** Starts the task.
- Stop** Stops the task immediately.
- Set Terminate** Stops the task the next time it checks its terminate flag. This is the recommended way to stop the task.
- Kill** Force stops a task. Use this option only if both the Set Terminate and Stop options could not stop the task. This usually indicates a task is caught in an infinite loop and is not reading its terminate flag.

If you use this option to stop a task and if the task had the real-time database locked at the time you stopped it, no other tasks will be able to access the real-time database until the lock is turned off.

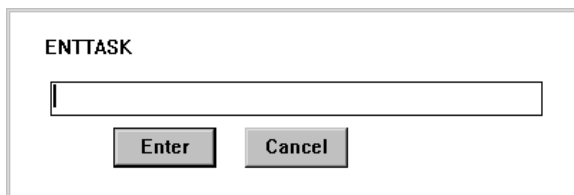
For Windows NT and Windows 95, re-boot the computer.

For OS/2, run the FLSHM utility with the `-d` parameter to correct this condition. Refer to “FactoryLink Utilities” on page 361 for details on using FLSHM.

For UNIX, FactoryLink monitors for this condition and attempts to unlock the real-time database.

- **USING RUN-TIME MONITOR**
- *Monitoring FactoryLink Processes*
-
-

Find Finds a task in the process list. When you choose Find, you are prompted for the task you want to find.

A screenshot of a dialog box titled "ENTTASK". It features a single-line text input field. Below the input field are two buttons: "Enter" and "Cancel". The dialog box has a standard Windows-style border.

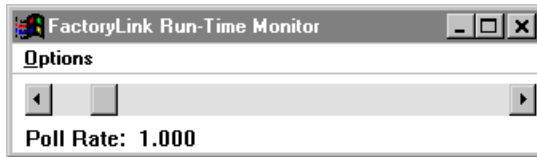
Enter the name of the executable. This locates the first occurrence of the specified FactoryLink task in the current process list window.

Exit Exits the process list window and returns to the FactoryLink Run-Time Monitor panel.

USING RUN-TIME MONITOR COMMANDS

RTMON provides a window to the operating system in order to execute commands that aid in developing, testing, or debugging the application. Perform the following steps to open this window.

- 1 Choose Options from the Run-Time Monitor panel.



The following pulldown menu is displayed.



- 2 Choose Command Input from the Options pulldown menu. The Monitor Commands panel is displayed.



- **USING RUN-TIME MONITOR**
- *Using Run-Time Monitor Commands*
-
-

The appearance of this window varies depending on the operating system. When entering commands in the Commands window, use the following guidelines.

- To recall previous commands, use the ↑ and ↓ keys.
- To scroll through the output, use the scroll bar.
- To execute a command in real-time mode, press Enter.
- To exit the Commands window, choose Exit from the Monitor Commands Options pulldown menu. This returns you to the Run-Time Monitor panel.

The information in the following pages describes how to use the commands available with the Monitor Commands window.

Terminating a Task

Use the `t` command to terminate a FactoryLink task. The syntax is

`t name`

where *name* is the name of the task to terminate as specified in the Task Name field of the System Configuration Information panel.

Exiting Commands Window

Use the `q` command to quit the session and exit the commands window. The syntax is

`q`

Accessing Help

Use the `?` or `h` command to open help. The syntax is

`? [cmd]`

or

`h [cmd]`

where *cmd* is the command you want help for. For example,

`?i`

opens help for the `i` command. If you do not specify *cmd*, help is provided for all commands.

Reading an Element

You can use several commands to read an element. These include:

Reading a Single Element

Use the `r` command to display the value for a single element. The syntax is

`r tag`

where *tag* identifies the element you want to read. This can either be the element name or element number. The element number is displayed in output when you read the element. Output is displayed on the screen unless you redirect it to a file using the `o` command.

For example, enter the following command to read the contents of analog element TEST.

```
r TEST
```

This generates the following output.

```
a (1)103 = 4108 x100C
```

where

- `a` is the data type of *tag*. In this case, the data type is analog.
- `(1)` is the segment where *tag* is located in memory.
- `103` is the number assigned to *tag*. This number can be used in command syntax in place of the *tag* name.
- `4108` is the decimal representation of the *tag*'s value.
- `x100c` is the hexadecimal representation of the *tag*'s value.

Reading One or More Elements

Use the `d` command to display the values for multiple elements of an array *tag* with a single command. The syntax is

`d tag count`

where

- tag* identifies the element to start with. This can either be the element name or element number. The element number is displayed in output when you read the element.

- **USING RUN-TIME MONITOR**
- *Using Run-Time Monitor Commands*
-
-

count identifies the number of elements in the array to display starting with the element identified by *tag*.

For example, to display the values of three elements in an array starting with *tag*, enter the following command.

```
d TEMP [4] 3
```

This generates the following output.

```
(01)0078: 15    5    2
```

where

(01) is the segment where *tag* is located.

0078 is the number assigned *tag*. This number can be used in command syntax in place of the *tag* name.

15, 5, 2 are the values of the three elements.

Press any key to stop the display from scrolling. Press any key except *s* to restart the scroll.

Press any key, then press *s* to stop the display.

Redirecting Read Output

Read output is sent to the terminal screen unless redirected to a file using the *o* command. The syntax is

```
o [filename]
```

where *filename* is the name of the file to receive the output. If you do not supply *filename*, output is sent to the terminal screen.

Writing to an Element

A number of commands write to an element. If the tag type is message, all text between the last character of the element name and the end of the line or semicolon (;) is written, including any leading or trailing spaces. Newlines and semicolons are not written to message elements.

Writing a Value to a Single Element

Use the `w` command to write a value to a single element. The syntax is

`w tag value`

where

- `tag` identifies the element to receive *value*. This can either be the element name or element number. The element number is displayed in output when you read the element.
- `value` identifies the value to write to *tag*. You can enter numerical values as hexadecimal numbers by preceding the value with `0x`.

For example, enter the following command to write a value of 3017 to TEST .

```
w TEST 3017
```

Force-Writing a Value to a Single Element

Use the `W` command to force write a value to a single element. A force write sets the Change-status flag to 1 (on). The syntax is

`W tag value`

where

- `tag` identifies the element to receive *value*. This can either be the element name or element number. The element number is displayed in output when you read the element.
- `value` identifies the value to write to *tag*. You can enter numerical values as hexadecimal numbers by preceding the value with `0x`.

- **USING RUN-TIME MONITOR**
- *Using Run-Time Monitor Commands*
-
-

Writing a Value to One or More Elements

Use the `s` command to write the same value to multiple elements in an array with a single command. The syntax is

s tag count value

where

- tag* identifies the element to start with. This can either be the element name or element number. The element number is displayed in output when you read the element.
- count* identifies the number of elements in the array to display starting with the element identified by *tag*.
- value* identifies the value to write to each tag. You can enter numerical values as hexadecimal numbers by preceding the value with 0x.

For example, to set the values of three elements to 100, beginning with TEMPSET, enter the following command.

```
s TEMPSET [0] 3 100
```

Force Writing a Value to One or More Elements

Use the `S` command to force-write the same value to multiple elements of an array tag with a single command. A force write sets the Change-status flag to 1 (on). The syntax is

S tag count value

where

- tag* identifies the element to start with. This can either be the element name or element number. The element number is displayed in output when you read the element.
- count* identifies the number of elements in the array to display starting with the element identified by *tag*.
- value* identifies the value to write to each tag. You can enter numerical values as hexadecimal numbers by preceding the value with 0x.

For example, enter the following command to set the values of three elements beginning with TEMPSET to 100.

```
S TEMPSET [0] 3 100
```

Working with Batch Files

This section discusses the commands used for creating and executing batch files. You can nest the execution of batch files which means you can have one batch file executed by another batch file. You can nest files four deep.

End a line in a batch file with a newline or semicolon (;).

Creating a Batch File

You can create a batch file by save keyboard input to a file. You can place a series of commands as well as comments in a batch file. Use the `f` command to create a batch file. The syntax is

```
f [filename]
```

where *filename* is the name of the file to receive the input. If you use the `f` command without the file name, input is not saved to a file and commands are executed as they are entered.

Adding Remarks to a Batch File

Use the `#` command to add a remark to a batch file. The syntax is

```
# [remark]
```

where *remark* is the text you do not want interpreted as a command. Terminate the remark with either a newline or a semicolon.

You should only use remarks in batch files.

Echoing a Comment in Output

Use the `c` command to echo comments in output. The syntax is

```
c [comment]
```

where *comment* is the text you want to be displayed in the output.

If you want the *comment* sent to a file, precede this command with the `o` command. Otherwise, the output is sent to the terminal screen.

- **USING RUN-TIME MONITOR**
- *Using Run-Time Monitor Commands*
-
-

Ending Current File

Place the `e` command at the end of the batch file to indicate the end of the file has been reached. The syntax is

`e`

If no more files are specified for execution or if the `e` command is executed from the keyboard, the Commands window exits.

Executing a Batch File Once

Use the `i` command to execute a batch file if you want to execute the batch file only once. Use the `l` command if you want to execute the same batch file multiple times. The syntax is

`i filename`

where *filename* is the name of the batch file to execute.

For example, enter the following command to take input from the file TESTOUT .

`i TESTOUT`

Executing a Batch File Multiple Times

Use the `l` command to execute the same batch file multiple times. The syntax is

`l count filename`

where

`count` is the number of times to execute the file.

`filename` is the name of the file to execute.

For example, enter the following command to read the commands from the file TESTOUT five times.

`l 5 testout`

- **USING RUN-TIME MONITOR**
- *Monitoring the Status of the Real-Time Database*
-
-

The following information is displayed in the window.

| | |
|-----------|---|
| Segments | <p>Number of segments in use in the real-time database. A segment is a finite block of space in the real-time database that contains elements of one data type.</p> <p>Elements for each data type are stored in separate segments. When a segment is full, the next time you define an element of that data type, the real-time database creates a new segment to contain the element.</p> |
| Digitals | Total number of digital elements defined. |
| Floats | Total number of floating-point elements defined. |
| Message | Total number of message elements defined. |
| Lock Id. | <p>Indicator of whether or not the real-time database is locked and, if so, which task has locked it.</p> <p>When a task wakes up to read from or write to the real-time database, it locks all other tasks out of the database. When the task has completed its operations, it unlocks the real-time database so another task can access it.</p> <p>Each task has its own lock ID number. If the real-time database is locked, this field displays the lock ID number of the task that locked it. If the real-time database is not locked, this field displays either -1 or 65,535 depending on the system in use.</p> |
| Memory | Amount of RAM used by the real-time database. The first number indicates the amount of RAM currently in use. The number in parentheses indicates the amount of RAM allocated to the real-time database. This includes available message space. |
| Analogs | Total number of analog elements defined. |
| Longs | Total number of longana elements defined. |
| Mailboxes | Total number of mailbox elements defined. |
| Link Id. | <p>Number identifying the next task that locks the real-time database.</p> <p>FactoryLink keeps a list of tasks waiting on the real-time database. Each task waiting for real-time database access is added to this list. The Link ID field displays only the task in that list with the highest priority.</p> |

The following fields contain only colors to indicate their status. Color indicators are: green = on; red = off.



Each column of colors represents a FactoryLink task. The first column on the left represents the first task that is displayed on the Run-Time Manager screen; the second column represents the second task; and so on. You can monitor the colors to see if all tasks seem to be active and changing states at regular intervals.

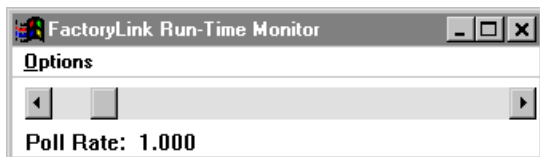
- Active** Indicates whether or not the task is active. Green indicates task is active. Red indicates task is not active.
- Sleep** Indicates whether or not the task is waiting for the kernel to wake it up to inform of an element's change in status. Green indicates waiting. Red indicates not waiting.
- Wait Lock** Indicates whether or not a task is waiting to lock the real-time database. When a task is reading from or writing to the real-time database, it locks all other tasks out of the database. Green indicates it is waiting. Red indicates it is not waiting.
- Wait Acc.** Indicates whether or not a task is waiting for access to the real-time database. Green indicates it is waiting. Red indicates it is not waiting.
- Wait Read** Indicates whether or not a task is waiting to perform a read operation. Green indicates it is waiting. Red indicates it is not waiting.
- Wait Write** Indicates whether or not a task is waiting to perform a write operation. Green indicates it is waiting. Red indicates it is not waiting.

- **USING RUN-TIME MONITOR**
- *Terminating All FactoryLink Tasks*
-
-

TERMINATING ALL FACTORYLINK TASKS

Perform the following steps to terminate all FactoryLink tasks in the current domain. Terminate all user domain tasks before terminating shared domain tasks.

- 1 Choose Options from the Run-Time Monitor panel.



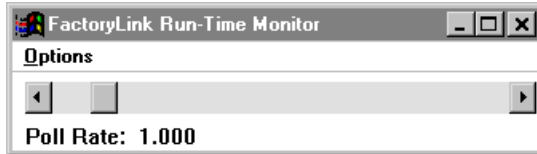
The following pulldown menu is displayed.



- 2 Choose Shutdown All from the Options pulldown menu.

EXITING RTMON

- 1 Choose Options from the Run-Time Monitor panel.



The following pulldown menu is displayed.



- 2 Choose Exit from the Options pulldown menu.

- **USING RUN-TIME MONITOR**

- *Run-time Manager Messages*

-
-

RUN-TIME MANAGER MESSAGES

If errors occur while using Run-time Manager, error messages are displayed on screen. This section lists these messages, describes their cause, and provides suggested actions.

Messages are described in two ways:

- Text message—Refer to “Text Messages” on page 348 for text messages.
- Error numbers—Refer to “Messages with Error Numbers” on page 354 for error number messages.

Text Messages

Bad command *number*

Cause: An invalid command was written to the global element COMMAND.

Action: Verify the external process' compatibility with FactoryLink.

Bad file size for *filename*

Cause: The FLAPP/CT/TYPE.CT file has been damaged.

Action: Delete the file and re-start the application to rebuild the file.

Bad index in file *filename*

Cause: Unable to read the index of the FLAPP/CT/TYPE.CT file.

Action: Delete the file and re-start the application to rebuild the file.

Can't convert subgroup table *name*

Cause: Tried to convert an older SPC application.

Action: See the conversions instructions in the Release Notes.

Can't create appl. *filename* error *error number*

Cause: Tried to start an application already started.

Action: No action required.

Can't open file *filename*

Cause: The disk may be full, or this may be a log file opened by another process.

Action: Delete unnecessary files. If this error occurs often, additional disk space may be required.

Can't read options file *FL.OPT*

Cause: FL.OPT has been damaged.

Action: Verify the contents of FL.OPT by running UKEY. Copy flnew.opt to fl.opt.

Can't start process *process name*

Cause: The binary file may not exist, may not be executable, or the filename in the EXECUTABLE FILE field in the System Configuration Table may be incorrect. Under some system platforms this may indicate an insufficient number of processes.

Action: Verify the correct file name in the System Configuration Table or see the *FactoryLink Fundamentals*.

Can't stop process *process name*

Cause: The process may already have stopped.

Action: Attempt to stop the process manually if it has not stopped.

Client *processes* failed to start

Cause: One or more processes could not be started.

Action: See the error messages for the particular process.

Client *processes* failed to stop

Cause: The process may already have stopped.

Action: Attempt to stop the process manually if it has not stopped.

Directory *directory name* does not exist

Cause: Can't find FLAPP directory specified by the environment variable.

Action: Set FLAPP to a valid application directory.

- **USING RUN-TIME MONITOR**

- *Run-time Manager Messages*

-
-

Directory *directory name* is not a valid FactoryLink directory

Cause: FLAPP does not have a valid subdirectory structure.

Action: (1) Check the FLAPP directory. Add the missing subdirectories.
(2) Set FLAPP to a valid application directory.

Domain *domain name* can only have one instance

Cause: A “parent” domain has been configured as having more than one instance.

Action: A “parent” domain may only have one instance. Open the Domain Element List and change the entry in the #INST field to 1.

Domain *domain name* isn’t in the domain CT

Cause: The specified domain does not exist in the domain .CT.

Action: The domain name may have been entered incorrectly or may not exist in the Domain Element List. Verify the domain exists in the Domain Element List and has been entered correctly.

Environment Tag *element name* has an invalid type

Cause: The wrong data type was specified for the field.

Action: Enter the correct data type for this field.

FactoryLink initialization failed

Cause: FactoryLink system failed to initialize.

Action: Ensure the master key is present and properly connected. Run KEYINST and FLKEYVAL.

FactoryLink system monitoring failed

Cause: The system was unable to start the monitor task. The system may not contain sufficient memory.

Action: Stop unnecessary processes.

Grace period has expired. Software must be registered

- Cause: FactoryLink was installed without registering it within the 10 day grace period.
- Action: Run FLKEYVAL and follow the instructions for registration.

Kernel initialization failed

- Cause: FactoryLink was already initialized. FactoryLink was already running when start-up was attempted.
- Action: No action required.

Monitor tag array definition failed

- Cause: A System Configuration Table monitor element (Start Trigger, Task Status, or Task Message) may be undefined or defined incorrectly for one of the tasks being started.
- Action: Open the System Configuration Table and define the elements correctly.

Number of defined proc *process number* more than max *maximum number*

- Cause: More than 31 processes were started.
- Action: Start fewer processes.

Out of RAM

- Cause: No more memory is available.
- Action: Close any unnecessary windows or programs. Add more memory to the system if this error occurs often.

Output tag array definition failed

- Cause: A System Configuration Table output element (Display Status, Display Name, or Display Description) may be undefined or defined incorrectly for one of the tasks being started.
- Action: Open the System Configuration Table and define the elements correctly.

- **USING RUN-TIME MONITOR**

- *Run-time Manager Messages*

-
-

Process *process name* may not have started

Cause: The process failed to become active. It may not have been able to register with FactoryLink.

Action: Verify the filename in the EXECUTABLE FILE field in the System Configuration Table. Verify the process' compatibility with FactoryLink.

Process *process name* may not have stopped

Cause: The process may already have stopped.

Action: Attempt to stop the process manually if it has not stopped.

Read failed on file *filename*

Cause: A read operation on the named file failed.

Action: Verify configuration table entries, communication parameters, hardware identification information, and electrical connections.

Read header failed on file *filename*

Cause: The .CT file may be damaged.

Action: Delete the .CT file in the FLINK/ct directory and re-start the application to rebuild the file.

Real-time database isn't initialized

Cause: FactoryLink is starting or shutting down.

Action: Start FactoryLink again.

Real-time database for *application* doesn't exist

Cause: An attempt was made to start a USER domain when the SHARED domain did not exist.

Action: Start the SHARED domain.

Run-time Manager: *errno* = *error number*

Cause: The Run-time Manager was awakened when there was nothing for it to do. (Internal error.)

Action: No action required.

Run-time Manager CT processing failed

Cause: One or more of the elements ARGUMENT, COMMAND, PASSWORD, SHUTDOWN, or STARTUP is not defined in the GLOBAL.CDB file. The file has been damaged or installed incorrectly.

Action: Contact Customer Support.

Run-time Manager failed to start

Cause: The Run-time Manager must be the first task started. The Start Order number for RUNMGR must be 0; no other task may have a zero for a Start Order number.

Action: Check the System Configuration Table to verify the Start Order.

Run-time Manager failed to stop

Cause: The process may already have stopped.

Action: Attempt to stop the process manually if it has not stopped.

Run-time Manager is already running

Cause: A copy of the Run-time Manager is already running for the DOMAIN and USER name specified.

Action: Change the DOMAIN or USER name.

Software has not been enabled

Cause: An error has occurred with the fl.key file in the opt directory.

Action: 1. Run KEYINST and if an invalid or archaic agreement is found, re-enter the configuration information.
2. Run FLKEYVAL and follow the instructions for registration.

Write error on log file *filename*

Cause: The disk may be full.

Action: Delete unnecessary files.

- **USING RUN-TIME MONITOR**

- *Run-time Manager Messages*

-
-

Messages with Error Numbers

Two Run-time Manager error messages display error numbers within the message:

Can't create appl. *filename* error *error number*
Run-time Manager: *errno* = *error number*

The error numbers in these messages are generated by the FactoryLink real-time database. Each error number has a corresponding error keyword. Even though this keyword is not displayed in the error message, knowing the error keyword helps Customer Support engineers identify the cause of the problem.

1 FLE_INTERNAL

Cause: Internal Error.

Action: See “Correcting Internal Errors” at the end of this section.

2 FLE_OUT_OF_MEMORY

Cause: There is not enough RAM available.

Action: Add RAM to the system.

4 FLE_NO_FLINK_INIT

Cause: The real-time database has not been created or cannot finish initializing.

Action: Shut down FactoryLink and re-start it.

5 FLE_NO_PROC_INIT

Cause: One of the following conditions caused this error:
The task name of the specified task has not been entered in the System Configuration Table.

Action: Perform the corresponding action:
Enter the task name in the Task Name field of the System Configuration Table.

Cause: The task is already running.

Action: Nothing—the task will continue to run.

Cause: The indicated task is not enabled on the FactoryLink key.

Action: Contact your FactoryLink sales representative to obtain the proper key and/or option.

7 FLE_BAD_ARGUMENT

Cause: Internal error—a task tried to pass an invalid argument to the real-time database.

Action: See “Correcting Internal Errors” at the end of this section.

9 FLE_BAD_TAG

Cause: The .CTs need to be rebuilt.

Action: Rebuild the .CTs by running CTGEN.

10 FLE_NULL_POINTER

Cause: Internal error—a task gave out a null pointer.

Action: See “Correcting Internal Errors” at the end of this section.

12 FLE_PROC_TABLE_FULL

Cause: 31 tasks are running in the chosen domain. Only 31 tasks can run in each domain at once.

Action: No action required—do not try to run more than 31 tasks per domain at a time.

13 FLE_BAD_PROC_NAME

Cause: One of the following conditions caused this error:
The .CTs need to be rebuilt.

Action: Perform the corresponding action:
Run CTGEN to rebuild the .CTs. If this does not solve the problem, proceed to step 2.

Cause: A task name used in the application has not been entered in the Task Name field of the System Configuration Table.

Action: Open the System Configuration Table from the Configuration Manager Main Menu and ensure the names of all tasks in the application are present in the Task Name field.

Cause: The requested task is not running.

- **USING RUN-TIME MONITOR**

- *Run-time Manager Messages*

-
-

Action: Ensure you started all tasks needed to run the application.

14 FLE_BAD_USER_NAME

Cause: The FLUSER environment variable is not set.

Action: Set the environment variables; re-start FactoryLink.

22 FLE_ALREADY_ACTIVE

Cause: The task is already running.

Action: None-do not start a task already running.

23 FLE_NOT_LOCKED

Cause: Internal error—a task tried to unlock the real-time database without having locked it first.

Action: See “Correcting Internal Errors” at the end of this section.

24 FLE_LOCK_FAILED

Cause: Internal error—a task used an invalid Task ID to lock the real-time database. Therefore, the task did not successfully lock the real-time database or perform its function.

Action: See “Correcting Internal Errors” at the end of this section.

25 FLE_LOCK_EXPIRED

Cause: A task has kept the real-time database locked longer than the kernel allows.

Action: None—currently, the kernel allows tasks to lock the real-time database for as long as necessary. Therefore, the lock time will not expire.

26 FLE_WAIT_FAILED

Cause: Internal error—while trying to do a change-wait on an element, a task sent an invalid Task ID to the real-time database.

Action: See “Correcting Internal Errors” at the end of this section.

28 FLE_QSIZE_TOOBIG

Cause: Internal error—a task attempted to attach a queue to an element, but there was not enough memory.

Action: See “Correcting Internal Errors” at the end of this section.

29 FLE_QSIZE_CHANGED

Cause: Internal error—a task attempted to attach a queue to an element, but a queue of a different size was already attached.

Action: See “Correcting Internal Errors” at the end of this section.

30 FLE_NO_TAG_LIST

Cause: This error only occurs on custom tasks written using the FactoryLink PAK. If the custom task was set up to access elements by name instead of by ID number, this error can occur if the element list has not been defined.

Action: Use the API function FL_SET_TAG_LIST to define the element list. Then, re-start FactoryLink. Refer to the *Programmer's Access Kit (PAK)* manual for information about L_SET_TAG_LIST.

31 FLE_TAG_LIST_CHANGED

Cause: This message can be displayed only if an application has been set up so one task monitors another task's element list. If one task modifies the other task's element list, the task that modified the list will return this message, thus informing the task that its list has been modified.

Action: No action required.

32 FLE_WAKEUP_FAILED

Cause: Internal error—The real-time database tried unsuccessfully to wake a task waiting for the value of a particular element to change.

Action: See “Correcting Internal Errors” at the end of this section.

- **USING RUN-TIME MONITOR**

- *Run-time Manager Messages*

-
-

33 FLE_NO_SIGNALS

Cause: If a task calls the FL_RECV_SIG function to find out if any signals have been sent to it, the FL_RECV_SIG function returns a message. If no signals have been sent to the task, FL_RECV_SIG returns this message.

Action: No action required.

34 FLE_SIGNALLED

Cause: The real-time database sends this message to inform a sleeping task that the task has received a signal, not a change in value of the element for which it is waiting.

Action: No action required.

35 FLE_NOT_MAILBOX

Cause: One of the following conditions caused this error:
The .CTs need to be rebuilt.

Action: Perform the corresponding action:
Run CTGEN to rebuild the .CTs. If this does not solve the problem, proceed to step 2.

Cause: While configuring a task, you entered an element of a data type other than MAILBOX in a field that requires a MAILBOX element.

Action: Open the configuration panels from the Main Menu of the task containing the error . Ensure the data types of the elements in all fields requiring MAILBOX elements, are defined as MAILBOX.

36 FLE_NO_MESSAGES

Cause: If through the PAK_QUERY_MBX function, a task requests to view a message whose queue number falls outside the range of available messages, the QUERY_MBX function returns this message. For example, if a task requests to view the third message waiting in the queue, and only two messages are in the queue, then the task is requesting to view a message outside the range of available messages.

Action: No action required.

37 FLE_ACCESS_DENIED

Cause: If a task tries to read a mailbox element that it is not allowed to read, the real-time database returns this error. If the task is a custom task developed using PAK, the programmer may not have set up ownership of the mailbox element by the task requesting to read from it.

Action: Use the PAK function FL_SET_OWNER_MBX to establish the task as owner of the mailbox element.

Cause: Internal error.

Action: See “Correcting Internal Errors,” at the end of this section.

41 FLE_APP_EXISTS

Cause: A real-time database with the same FLNAME but a different number of users or elements already exists.

Action: Either shut down the running real-time database or set FLNAME with a different name.

42 FLE_NO_FLINK_RTDB

Cause: A user domain was started for a real-time database for which the shared domain has not been started.

Action: Start the shared domain first.

- **USING RUN-TIME MONITOR**

- *Run-time Manager Messages*

-
-

Correcting Internal Errors

Internal errors are generally caused by one of the tasks in the system in use and not by your application. For this reason, we recommend you use the following guidelines to correct an internal error:

1. Try to determine which task is sending the error by shutting down FactoryLink, restarting it, and starting each task, one at a time.
2. Write down any error messages displayed on the Run-time Manager screen and their corresponding tasks. (The task having the problem may generate a seemingly unrelated error message.)
3. Contact the supplier of the task in error if the task in error is an external task.
4. Contact Customer Support if the task in error is a FactoryLink task.
5. Contact Customer Support if the task cannot be identified.

FactoryLink Utilities

FactoryLink provides several utilities for general maintenance. These are listed below.

- FLNEW—Creates a new FactoryLink application with a baseline configuration.
- FLREST—Restores a saved FactoryLink application.
- FLSETLNG—Sets FactoryLink applications to run in a specified language.
- FLTEST—Use for testing the installation
- FLDEMO—Use for application development ideas
- FLCONV—Converts an application from its current environment to a new environment when upgrading versions or migrating among operations.
- FLSAVE—Saves FactoryLink application to diskette or to another directory on the hard drive.
- CTGEN—Rebuilds CT files.
- CDBLIST—Use to view the binary CDB files in ASCII representation.
- CTLIST—Use to view the binary CT files in ASCII representation.
- DBCHK—Fixes FactoryLink database index files
- KEYINST—Installs FactoryLink license with specified options.
- FLKEYVAL— Enables FactoryLink software subsequent to installation.
- FLSHM—A memory display utility that lists existing FactoryLink real-time database memory areas. You can use this information to clean up memory if FactoryLink aborts.
- UKEY—Displays license options.
- BH_SQL—Refer to *FactoryLink Configuration Guide* for details for using this utility to maintain FactoryLink dBASE IV database files.

This chapter describes how to use these utilities. Command line formats and case may vary from one platform to the next. Open help using the question mark at the command line prompt to get details for your specific platform.

- **FACTORYLINK UTILITIES**
- *FLNEW*
-
-

FLNEW

After installing FactoryLink for the first time, you must create a new application using FLNEW to serve as the base upon which to build a FactoryLink application. This utility creates the necessary subdirectories.

FLNEW overwrites any FactoryLink application already existing in the destination directory. To preserve an existing application, back it up by performing an FLSAVE before executing the FLNEW utility or ensure you are creating the new application in an empty or new directory.

FLNEW supplies a set of libraries consisting of a collection of application components used to perform graphical, logical, process, and communication operations. Refer to the *FactoryLink Release Notes* for details on the most current libraries.

FLNEW contains the following pre-defined tags:

Table 109-1

| Tagname | Domain | Type | Description |
|--------------|--------|---------|--|
| ALC_ACK | USER | DIGITAL | Acknowledge most recent Alarm |
| ALC_ARCHIVE | SHARED | DIGITAL | Alarm Supervisor Control Archive |
| ALC_BANNER | SHARED | DIGITAL | Alarm Supervisor Control Banner |
| ALC_EXEC | SHARED | DIGITAL | Alarm Supervisor Control Execute |
| ALC_GROUP | SHARED | DIGITAL | Alarm Supervisor Control Group |
| ALC_GRPACK | USER | DIGITAL | Alarm Supervisor Control Group Acknowledge |
| ALC_LOGBK | USER | DIGITAL | Alarm logbook trigger |
| ALC_PGDN | SHARED | ANALOG | Alarm Supervisor Control Page down |
| ALC_PGUP | SHARED | ANALOG | Alarm Supervisor Control Page up |
| ALC_PRINT | SHARED | DIGITAL | Alarm Supervisor Control Print |
| ALC_RETURN | SHARED | DIGITAL | Alarm Supervisor Control Enter |
| ALC_SCROLL | SHARED | ANALOG | Alarm Supervisor Control Scroll |
| ALC_SDN | SHARED | ANALOG | Alarm Supervisor Control Scroll down |
| ALC_SELACK | USER | DIGITAL | Alarm Supervisor Control Selection Acknowledge |
| ALC_SORT | SHARED | DIGITAL | Alarm Supervisor Control Sort |
| ALC_SUP | SHARED | ANALOG | Alarm Supervisor Control Scroll up |
| ALD_ACTCNT | SHARED | ANALOG | Current Alarm Supervisor active alarm count |
| ALD_AUDCNT | SHARED | ANALOG | Current Alarm Supervisor audible alarm count |
| ALD_BANNER | SHARED | MESSAGE | Alarm Supervisor banner mode |
| ALD_GROUP | USER | MESSAGE | Current Alarm Supervisor group |
| ALD_LOG_TEXT | USER | MESSAGE | Alarm logbook text field |
| ALD_PRINTER | SHARED | MESSAGE | Current Alarm Supervisor print mode |

- **FACTORYLINK UTILITIES**
- *FLNEW*
-
-

Table 109-1

| Tagname | Domain | Type | Description |
|---------------|--------|---------|---|
| ALD_ROW | USER | ANALOG | Alarm Supervisor selected row tag |
| ALD_SORT | USER | MESSAGE | Current Alarm Supervisor sort mode |
| ALM_BANNER | USER | MESSAGE | Alarm Supervisor banner message |
| ALM_BANR_BG | USER | ANALOG | ALOG banner background color |
| ALM_BANR_BL | USER | ANALOG | ALOG banner line blink attribute |
| ALM_BANR_FG | USER | ANALOG | ALOG banner foreground color |
| ALOGBCOLOR | USER | ANALOG | Alarm Supervisor Text Background Color |
| ALOGBLINK | USER | ANALOG | Alarm Supervisor Text Blink Attribute |
| ALOGFCOLOR | USER | ANALOG | Alarm Supervisor Text Foreground Color |
| ALOGMBX_S | SHARED | MAILBOX | Alarm Supervisor Mailbox Tags |
| ALOGRFSH_S | SHARED | DIGITAL | Alarm Supervisor Viewer Screen Refresh Triggers |
| ALOGSHMEM_S | SHARED | LONGANA | Alarm Supervisor Shared Memory Sizes |
| ALOGTEXT | USER | MESSAGE | Alarm Supervisor Screen Text |
| ALOGVER_S | SHARED | ANALOG | Alarm Supervisor Version Information |
| ALOG_MBX | SHARED | MAILBOX | Alarm logger mailbox |
| A_DAY | SHARED | ANALOG | Day of the month |
| A_DOW | SHARED | ANALOG | Day of the week |
| A_DOY | SHARED | ANALOG | Day of the year |
| A_HOUR | SHARED | ANALOG | Number of hours past midnight |
| A_MIN | SHARED | ANALOG | Number of minutes past the hour |
| A_MONTH | SHARED | ANALOG | Month of the year |
| A_SEC | SHARED | ANALOG | Number of seconds past the minute |
| A_YEAR | SHARED | ANALOG | Current Year |
| BROWSEHISTMBX | SHARED | MAILBOX | Mailbox for BROWSER process |

Table 109-1

| Tagname | Domain | Type | Description |
|-----------------|--------|---------|---|
| BROWSEHISTMBX_U | USER | MAILBOX | Mailbox for BROWSER process |
| CONNSRVACTIVE | SHARED | ANALOG | Number of brokered services that are active |
| CONNSRVINIT | SHARED | DIGITAL | Connection server initialization flare |
| CONNSRVMBX | SHARED | MAILBOX | Connection server receive mailbox |
| CONNSRVTOTAL | SHARED | ANALOG | Total number of services being brokered |
| DALOGACKMBX | SHARED | MAILBOX | Distributed AL_LOG mailbox for acknowledged alarms |
| DALOGRCVMBX | SHARED | MAILBOX | Distributed AL_LOG Receive mailbox for Historian communications |
| DALOGRCVMBX_U | USER | MAILBOX | Distributed AL_LOG Receive mailbox for Historian communications |
| DALOGVIEWMBX | SHARED | MAILBOX | Distributed AL_VIEW mailbox for alarm communications |
| DALOGVIEWMBX_U | USER | MAILBOX | Distributed AL_VIEW mailbox for alarm communications |
| DATASRVMBX | USER | MAILBOX | Data server receive mailbox |
| DATE | SHARED | MESSAGE | Date (day MM/DD/YYYY) |
| DATE0 | SHARED | MESSAGE | Date (day MM/DD/YYYY) |
| DATETIME | SHARED | MESSAGE | Date and Time (day MM/DD/YYYY HH:MM:SS) |
| DBLOGHISTMBX | SHARED | MAILBOX | DBLOG Receive mailbox for Historian communications |
| DBLOGHISTMBX_U | USER | MAILBOX | DBLOG Receive mailbox for Historian communication |
| DPLOGHISTMBX | SHARED | MAILBOX | Data Point Logger Receive mailbox for Historian communications |
| FLAPP_S | SHARED | MESSAGE | Application directory |
| FLAPP_U | USER | MESSAGE | Application directory |
| FLDOMAIN_S | SHARED | MESSAGE | Domain Name |

- **FACTORYLINK UTILITIES**
- *FLNEW*
-
-

Table 109-1

| Tagname | Domain | Type | Description |
|---------------------|--------|---------|--|
| FLDOMAIN_U | USER | MESSAGE | Domain Name |
| FLINK_S | SHARED | MESSAGE | Directory for application programs |
| FLINK_U | USER | MESSAGE | Directory for application programs |
| FLLANSIG | SHARED | DIGITAL | FLLANSND system interval trigger |
| FLLANSNDBOX | SHARED | MAILBOX | Mailbox for FLLANSND process |
| FLNAME_S | SHARED | MESSAGE | Application Name |
| FLNAME_U | USER | MESSAGE | Application Name |
| FLOPERATOR_S | SHARED | MESSAGE | Application Operator |
| FLOPERATOR_U | USER | MESSAGE | Application Operator |
| LSECEVENTUSER_S | SHARED | MESSAGE | User Name for Application Security Event |
| LSECEVENTUSER_U | USER | MESSAGE | User Name for Application Security Event |
| FLSECEVENT_S | SHARED | MESSAGE | Application Security Event |
| FLSECEVENT_U | USER | MESSAGE | Application Security Event |
| FLUSER_S | SHARED | MESSAGE | Application User |
| FLUSER_U | USER | MESSAGE | Application User |
| FL_MAINWINNAME_U | USER | MESSAGE | FL_MAIN window screen tag |
| FL_TESTWINNAME | USER | MESSAGE | FL_TEST window screen name tag |
| GRAPHCONNTYPE | USER | ANALOG | Graph connection and security type |
| GRAPHMBX | SHARED | MAILBOX | Graphics Input Mailbox |
| GRAPHMBX_U | USER | MAILBOX | Graphics Input Mailbox |
| HISTOGRAM_STTIME | USER | LONGANA | |
| HISTOGRAM_Y_MAX_VAL | USER | FLOAT | |
| HISTOGRAM_Y_MIN_VAL | USER | FLOAT | |
| MAIN_WINNAME | SHARED | MESSAGE | FL_TEST window name for shared iml |

Table 109-1

| Tagname | Domain | Type | Description |
|---------------------|--------|---------|---|
| NULL_MSG | SHARED | MESSAGE | Blank message tag |
| NULL_MSG_U | USER | MESSAGE | Blank message tag |
| RTMARG | SHARED | ANALOG | Run-Time Manager Argument |
| RTMARG_U | USER | ANALOG | Run-Time Manager Argument |
| RTMCMD | SHARED | ANALOG | Run-Time Manager System Command |
| RTMCMD_U | USER | ANALOG | Run-Time Manager System Command |
| RTMPWD | SHARED | MESSAGE | Run-Time Manager Password |
| RTMPWD_U | USER | MESSAGE | Run-Time Manager Password |
| SECDAY | SHARED | LONGANA | Number of seconds past midnight |
| SECTIME | SHARED | LONGANA | Number of seconds since \Start of Time\ |
| SECYEAR | SHARED | LONGANA | Number of seconds past January 1 |
| SHUTDOWN 00:00:00 | SHARED | DIGITAL | Run-Time Manager Shutdown flag |
| SHUTDOWN_U | USER | DIGITAL | Run-Time Manager Shutdown flag |
| SPCCHART_STTIME | USER | LONGANA | |
| SPCCHART_Y_MAX_VAL | USER | FLOAT | |
| SPCCHART_Y_MIN_VAL | USER | FLOAT | |
| SPCDATAMBX_S | SHARED | MAILBOX | PowerSPC Data Task Receive Mailbox |
| SPCDATATRIG_S | SHARED | DIGITAL | PowerSPC Data Task Initialization Indicator |
| SPCDATA_APPDISABLE | SHARED | DIGITAL | Tag to toggle disable/enable of Power SPC processes |
| SPCDATA_APPDISABLE0 | SHARED | DIGITAL | |
| SPCGMBX | SHARED | MAILBOX | SPC Graphics input Mailbox |
| SPCGMBX_U | USER | MAILBOX | SPC Graphics input Mailbox |
| SPCGRPHMBX_S | SHARED | MAILBOX | PowerSPC Graphics Task Receive Mailbox |
| SPCGRPHMBX_U | USER | MAILBOX | PowerSPC Graphics Task Receive Mailbox |

- **FACTORYLINK UTILITIES**
- *FLNEW*
-
-

Table 109-1

| Tagname | Domain | Type | Description |
|---------------|--------|---------|---|
| SPRCVMBX | SHARED | MAILBOX | SPC Receive Mailbox for Database access |
| SPCRCVMBX_U | USER | MAILBOX | SPC Receive Mailbox for Database access |
| SPOOLREQ | SHARED | MESSAGE | Print Spooler Request |
| SPOOLREQ_U | USER | MESSAGE | Print Spooler Request |
| SPOOLRPLY | SHARED | MESSAGE | Print Spooler Reply |
| SPOOLRPLY_U | USER | MESSAGE | Print Spooler Reply |
| SPRGMBX | SHARED | MAILBOX | SPR graphics input mailbox |
| SPRGMBX_U | USER | MAILBOX | SPR graphics input mailbox |
| SPRRCVMBX | SHARED | MAILBOX | SPR receive mailbox for database access |
| SPRRCVMBX_U | USER | MAILBOX | SPR receive mailbox for database access |
| SPRSNDMBX | SHARED | MAILBOX | SPR historian mailbox |
| SPVRCVMBX | SHARED | MAILBOX | SPC View Mailbox for Database access |
| SPVRCVMBX_U | USER | MAILBOX | SPC View Mailbox for Database access |
| STARTUP | SHARED | DIGITAL | Run-Time Manager Startup flag |
| STARTUP_U | USER | DIGITAL | Run-Time Manager Startup flag |
| TASKDESC_S | SHARED | MESSAGE | Shared Task Description |
| TASKDESC_U | USER | MESSAGE | User Task Description |
| TASKDSTATUS_S | SHARED | MESSAGE | Shared Task Status Word |
| TASKDSTATUS_U | USER | MESSAGE | User Task Display Status Message |
| TASKMESSAGE_S | SHARED | MESSAGE | Shared Task Display Message |
| TASKMESSAGE_U | SHARED | MESSAGE | User Task Message |
| TASKNAME_S | SHARED | MESSAGE | Shared Task Name |
| TASKNAME_U | USER | MESSAGE | User Task Name |
| TASKSTART_S | SHARED | DIGITAL | Shared Task Start Trigger |

Table 109-1

| Tagname | Domain | Type | Description |
|-----------------|--------|---------|--|
| TASKSTART_U | USER | DIGITAL | User Task Start Trigger |
| TASKSTATUS_S | SHARED | ANALOG | Shared Task Status Value |
| TASKSTATUS_U | USER | ANALOG | User Task Status Value |
| TIME | SHARED | MESSAGE | Time (HH:MM:SS) |
| TIME0 | SHARED | MESSAGE | Time (HH:MM:SS) |
| TOPWINDOW_U | USER | MESSAGE | Current Top Window Name |
| TRENDBHISTMBX | SHARED | MAILBOX | TREND Receive mailbox for Historian communications |
| TRENDBHISTMBX_U | USER | MAILBOX | TREND Receive mailbox for Historian communication |
| TRENDBMX | SHARED | MAILBOX | Trend Input Mailbox |
| TRENDBMX_U | USER | MAILBOX | Trend Input Mailbox |
| VBLOGDISABLE_S | SHARED | DIGITAL | PowerVB debug logging disable |
| VBLOGDISABLE_U | USER | DIGITAL | PowerVB debug logging disable |
| YYMMDD | SHARED | MESSAGE | Date (YYMMDD) |
| alog_ana1 | SHARED | ANALOG | Alarm Supervisor analog >5 test |
| alog_ana1_limit | SHARED | ANALOG | Alarm Supervisor ana1 limit value |
| alog_ana1_stat | SHARED | ANALOG | Alog status word |
| alog_ana2 | SHARED | ANALOG | Alarm Supervisor analog < -5 alarm |
| alog_ana2_stat | SHARED | ANALOG | Alog status word |
| alog_dig1 | SHARED | DIGITAL | Alarm Supervisor digital test tag |
| alog_dig1_stat | SHARED | ANALOG | Alog status word |
| alog_flp1 | SHARED | FLOAT | Alarm Supervisor <> 0 test tag |
| alog_flp1_stat | SHARED | ANALOG | Alog status word |
| alog_lana1 | SHARED | LONGANA | Alarm Supervisor >= 12 test tag |

- **FACTORYLINK UTILITIES**
- *FLNEW*
-
-

Table 109-1

| Tagname | Domain | Type | Description |
|-----------------|--------|---------|----------------------------------|
| alog_lana1_stat | SHARED | ANALOG | Alog status word |
| alog_lana2 | SHARED | LONGANA | Alarm Supervisor <= -12 test tag |
| alog_lana2_stat | SHARED | ANALOG | Alog status word |
| alog_reset | SHARED | DIGITAL | Alarm Supervisor test values to0 |
| browsembx | SHARED | MAILBOX | Browser historian mailbox |
| brw_count | USER | ANALOG | Browse count data |
| brw_delete | USER | DIGITAL | Browser delete trigger |
| brw_move | USER | ANALOG | Browser move value |
| brw_position | USER | ANALOG | Browser position value |
| brw_row | USER | ANALOG | Browser current row value |
| brw_sec | USER | ANALOG | Browse seconds data |
| brw_select | USER | DIGITAL | Browser select trigger |
| brw_status | USER | MESSAGE | Browser action status word |
| brw_time | USER | MESSAGE | Browse time data |
| ctr_clear3 | SHARED | DIGITAL | counter clear tag for 3 |
| ctr_clear3_u | USER | DIGITAL | counter clear tag for 3 |
| ctr_dn_trig3 | SHARED | DIGITAL | counter down trigger for 3 |
| ctr_dn_trig3_u | USER | DIGITAL | counter down trigger for 3 |
| ctr_enable3 | SHARED | DIGITAL | counter enable tag for 3 |
| ctr_enable3_u | USER | DIGITAL | counter enable tag for 3 |
| ctr_neg_out3_u | USER | DIGITAL | counter negative output for 3 |
| ctr_neg_output3 | SHARED | DIGITAL | counter negative output for 3 |
| ctr_pos_out2_u | USER | DIGITAL | counter positive output for 2 |
| ctr_pos_out3_u | USER | DIGITAL | counter positive output for 3 |

Table 109-1

| Tagname | Domain | Type | Description |
|--------------------|--------|---------|-------------------------------------|
| ctr_pos_output2 | SHARED | DIGITAL | counter positive output for 2 |
| ctr_pos_output3 | SHARED | DIGITAL | counter positive output for 3 |
| ctr_up_trig1 | SHARED | DIGITAL | counter up clock trigger for 1 |
| ctr_up_trig2 | SHARED | DIGITAL | counter up trigger for 2 |
| ctr_uptrig3 | SHARED | DIGITAL | counter up trigger for 3 |
| ctr_up_trig3_u | USER | DIGITAL | counter up trigger for 3 |
| ctr_value1 | SHARED | ANALOG | counter current value for 1 |
| ctr_value1_u | USER | ANALOG | counter current value for 1 |
| ctr_value2 | SHARED | ANALOG | counter current value for 2 |
| ctr_value2_u | USER | ANALOG | counter current value for 2 |
| ctr_value3 | SHARED | ANALOG | counter current value for 3 |
| ctr_value3_u | USER | ANALOG | counter current value for 3 |
| demo_belt | USER | ANALOG | Demo screen belt |
| demo_belt_bot_xpos | USER | ANALOG | Demo screen belt bottom X position |
| demo_belt_top_xpos | USER | ANALOG | Demo screen belt top X position |
| demo_blue_botvalve | USER | DIGITAL | Demo screen blue pipe bottom valve |
| demo_blue_pipe | USER | ANALOG | Demo screen blue pipe level |
| demo_blue_pour | USER | DIGITAL | Demo screen blue pipe pouring |
| demo_blue_topvalve | USER | DIGITAL | Demo screen blue pipe top valve |
| demo_pick | USER | ANALOG | Demo screen picker |
| demo_pick_xpos | USER | ANALOG | Demo screen picker X position |
| demo_pick_ypos | USER | ANALOG | Demo screen picker Y position |
| demo_tank | USER | ANALOG | Demo screen tank level |
| demo_tank_level | USER | MESSAGE | Demo screen tank level output value |

- **FACTORYLINK UTILITIES**
- *FLNEW*
-
-

Table 109-1

| Tagname | Domain | Type | Description |
|----------------------|--------|---------|--|
| demo_tank_level_fg | USER | ANALOG | Demo screen tank level output foreground color |
| demo_tank_level_xpos | USER | ANALOG | Demo screen tank level output X position |
| demo_tank_xpos | USER | ANALOG | Demo screen tank X position |
| demo_tank_ypos | USER | ANALOG | Demo screen tank Y position |
| demo_trig | USER | DIGITAL | Demo screen math trigger |
| demo_wheel | USER | ANALOG | Demo screen belt wheel drive |
| demo_yel_botvalve | USER | DIGITAL | Demo screen yellow pipe bottom valve |
| demo_yel_pipe | USER | ANALOG | Demo screen yellow pipe level |
| demo_yel_pour | USER | DIGITAL | Demo screen yellow pipe pouring |
| demo_yel_topvalve | USER | DIGITAL | Demo screen yellow pipe top valve |
| edi_blk1 | USER | ANALOG | EDI block port 1 |
| edi_blk2 | SHARED | ANALOG | EDI block 2 tags |
| edi_blkwr1_state | SHARED | DIGITAL | EDI block write state for com1 |
| edi_blkwr1_trig | SHARED | DIGITAL | EDI block write trigger for com1 |
| edi_blkwr2_state | SHARED | DIGITAL | EDI block write state for com2 |
| edi_blkwr2_trig | SHARED | DIGITAL | EDI block write trigger for com2 |
| edi_test | SHARED | DIGITAL | EDI math table trigger |
| edi_usl1 | SHARED | ANALOG | EDI unsolicited block 1 |
| edi_usl2 | SHARED | ANALOG | EDI unsolicited block 2 |
| edi_write | SHARED | DIGITAL | EDI block write trigger |
| flfm_del1 | USER | DIGITAL | File Manager delete trigger |
| flfm_dir1 | USER | DIGITAL | File manager directory trigger |
| flfm_dir_rscrl | USER | ANALOG | FLFM remote DIR scroll tag |
| flfm_dir_scr11 | USER | ANALOG | File manager directory scroll tag |

Table 109-1

| Tagname | Domain | Type | Description |
|------------------|--------|---------|--|
| flfm_rdel | USER | DIGITAL | FLFM remote DEL command trigger |
| flfm_rdir | USER | DIGITAL | FLFM remote DIR trigger |
| flfm_remote_dir | USER | MESSAGE | FLFM remote directory name |
| flfm_remote_file | USER | MESSAGE | FLFM remote file name |
| flfm_remote_mess | USER | MESSAGE | FLFM remote command message |
| flfm_remote_stat | USER | ANALOG | FLFM remote command status value |
| flfm_rtext | USER | MESSAGE | FLFM remote command output text |
| flfm_rtype | USER | DIGITAL | FLFM remote TYPE command trigger |
| flfm_text | USER | MESSAGE | File Manager text array for D+T |
| flfm_type1 | USER | DIGITAL | File manager type trigger |
| flfm_type_rscr1 | USER | ANALOG | FLFM remote TYPE scroll value |
| flfm_type_scr11 | USER | ANALOG | File manager type scroll tag |
| float_change | USER | ANALOG | Float screen value change request value |
| float_value | USER | FLOAT | Float screen test value |
| help_file | USER | MESSAGE | FLFM help file name |
| help_scroll | USER | ANALOG | FLFM help screen scroll tag |
| help_text | USER | MESSAGE | FLFM help file type text tags |
| help_trig | USER | ANALOG | FLFM help file type command trigger |
| help_wanted | USER | DIGITAL | Help wanted for the current screen trigger |
| hist_spc | SHARED | MAILBOX | SPC to historian mailbox |
| histmbx | SHARED | MAILBOX | Historian mailbox for logging trend info |
| iml_call | SHARED | DIGITAL | IML procedure call test trigger |
| iml_const | SHARED | DIGITAL | IML constant trigger |
| iml_ctrl_trig | SHARED | DIGITAL | Iml control procedure trigger |

- **FACTORYLINK UTILITIES**
- *FLNEW*
-
-

Table 109-1

| Tagname | Domain | Type | Description |
|----------------|--------|---------|--|
| iml_gdeclare | SHARED | DIGITAL | IML test trigger for global declare var |
| iml_iftest | SHARED | DIGITAL | IML if test trigger/value |
| iml_ldeclare | SHARED | DIGITAL | IML local declare test trigger |
| iml_mathfun | SHARED | DIGITAL | IML math function test |
| iml_out | SHARED | MESSAGE | Iml information output tag |
| iml_out_u | USER | MESSAGE | Iml output message for test |
| iml_strfun | SHARED | DIGITAL | IML string function test trigger |
| iml_trig_s | SHARED | DIGITAL | Iml test proc trigger |
| iml_trig_u | USER | DIGITAL | Iml test proc trigger |
| iml_while | SHARED | DIGITAL | IML while structure test trigger |
| imlu_call | USER | DIGITAL | IML procedure call trigger |
| imlu_const | USER | DIGITAL | IML constant test trigger |
| imlu_gdeclare | USER | DIGITAL | IML global declare variable trigger |
| imlu_iftest | USER | DIGITAL | IML IF structure test trigger |
| imlu_ldeclare | USER | DIGITAL | IML local declare trigger |
| imlu_mathfun | USER | DIGITAL | IML math function test trigger |
| imlu_strfun | USER | DIGITAL | IML string function test trigger |
| imlu_while | USER | DIGITAL | IML while structure trigger |
| lan_begin | SHARED | DIGITAL | LAN begin connections trigger |
| lan_local_file | SHARED | DIGITAL | LAN local file creation done from report gen |
| lan_localname | SHARED | MESSAGE | LAN local name |
| lan_math | SHARED | DIGITAL | LAN timer trigger for sending values |
| lan_recv_ana | SHARED | ANALOG | LAN reveive analog tags |
| lan_recv_dig | SHARED | DIGITAL | LAN receive digital tags |

Table 109-1

| Tagname | Domain | Type | Description |
|-----------------|--------|---------|--|
| lan_recv_flp | SHARED | FLOAT | LAN receive float tags |
| lan_recv_lana | SHARED | LONGANA | LAN receive long analog tags |
| lan_recv_msg | SHARED | MESSAGE | LAN receive message tags |
| lan_remote_file | SHARED | DIGITAL | LAN remote file creation complete from rpt |
| lan_remotename | SHARED | MESSAGE | LAN remote name |
| lan_send_ana | SHARED | ANALOG | LAN send analog tags |
| lan_send_dig | SHARED | DIGITAL | LAN send digital tags |
| lan_send_flp | SHARED | FLOAT | LAN send float tags |
| lan_send_lana | SHARED | LONGANA | LAN send long analog tags |
| lan_send_msg | SHARED | MESSAGE | LAN send message tags |
| lan_test | SHARED | DIGITAL | LAN trigger for math |
| logtrig | SHARED | DIGITAL | Logger - timer trigger |
| persist_trig | SHARED | DIGITAL | Trigger for saving Persistence information |
| printscr | USER | DIGITAL | Print screen trigger for Rainbow screen |
| rcp_data | SHARED | FLOAT | Recipe data array |
| rcp_datetime | SHARED | MESSAGE | Recipe date and time |
| rcp_name | SHARED | MESSAGE | Recipe name |
| rcp_number | SHARED | ANALOG | Recipe number (also file number) |
| rcp_test | SHARED | DIGITAL | IML recipe procedure trigger |
| rcpdone | SHARED | DIGITAL | Recipe completion trigger |
| rcpfile | SHARED | ANALOG | Recipe file number |
| rcpread | SHARED | DIGITAL | Recipe read trigger |
| rcpstat | SHARED | ANALOG | Recipe status |
| rcpwrite | SHARED | DIGITAL | Recipe write trigger |

- **FACTORYLINK UTILITIES**
- *FLNEW*
-
-

Table 109-1

| Tagname | Domain | Type | Description |
|-------------------|--------|---------|--|
| rpt_begin1 | SHARED | DIGITAL | Report generator begin trigger |
| rpt_done1 | SHARED | DIGITAL | Report generator complete trigger |
| rpt_end1 | SHARED | DIGITAL | Report generator end trigger |
| rpt_operator | SHARED | MESSAGE | Report generator operator tag |
| rpt_repeat1 | SHARED | DIGITAL | Report generator repeat trigger |
| sec1 | SHARED | DIGITAL | one second timer trigger |
| spc_cause | USER | MESSAGE | SPC chart cursor cause code |
| spc_chart_cursor | USER | ANALOG | SPC graphics chart cursor tag |
| spc_charttype | USER | MESSAGE | SPC chart type tag |
| spc_cur_value | USER | FLOAT | SPC cursor calculated chart value |
| spc_cursor | USER | ANALOG | SPC chart cursor value |
| spc_datetime | USER | MESSAGE | SPC cursor value date and time |
| spc_freeze | USER | DIGITAL | SPC chart freeze tag |
| spc_lowcontrol | USER | FLOAT | SPC cursor lower control |
| spc_lowspec | USER | FLOAT | SPC cursor lower spec |
| spc_mode | USER | DIGITAL | SPC chart mode tag |
| spc_pan | USER | ANALOG | SPC pan value |
| spc_range_3sig | SHARED | DIGITAL | SPC range run-rule violation |
| spc_range_lowspec | SHARED | FLOAT | SPC range lower spec limit |
| spc_range_upspec | SHARED | FLOAT | SPC range upper spec limit |
| spc_sigma_3sig | SHARED | DIGITAL | SPC standard deviation 3-sigma violation |
| spc_sigma_lowspec | SHARED | FLOAT | SPC sigma lower spec limit |
| spc_sigma_upspec | SHARED | FLOAT | SPC sigma upper spec limit |
| spc_upcenter | USER | FLOAT | SPC cursor upper center line |

Table 109-1

| Tagname | Domain | Type | Description |
|------------------|--------|---------|-------------------------------------|
| spc_upcontrol | USER | FLOAT | SPC cursor upper control |
| spc_upspec | USER | FLOAT | SPC cursor upper spec |
| spc_xbar_3sig | SHARED | DIGITAL | SPC xbar 3 sigma run-rule violation |
| spc_xbar_lowspec | SHARED | FLOAT | SPC xbar lower spec limit |
| spc_xbar_upspec | SHARED | FLOAT | SPC xbar upper spec limit |
| spctrig | SHARED | DIGITAL | SPC sample trigger |
| spool_msg | SHARED | MESSAGE | Spooler message for device 1 |
| spr_center | SHARED | FLOAT | SPR center line |
| spr_histtrig | USER | DIGITAL | SPR histogram trigger |
| spr_lowcontrol | SHARED | FLOAT | SPR lower control limit |
| spr_recalc | SHARED | DIGITAL | SPR recalculation trigge |
| spr_recalc2 | SHARED | DIGITAL | SPR recalculation trigger for XBARS |
| spr_scenter | SHARED | FLOAT | SPR sigma center line |
| spr_slowcontrol | SHARED | FLOAT | SPR sigma lower control line |
| spr_supcontrol | SHARED | FLOAT | SPR sigma upper control limit |
| spr_upcontrol | SHARED | FLOAT | SPR upper control |
| spr_xcenter | SHARED | FLOAT | SPR xbar center line |
| spr_xlowcontrol | SHARED | FLOAT | SPR xbar lower control |
| spr_xupcontrol | SHARED | FLOAT | SPR xbar upper control |
| trendrepmbx | USER | MAILBOX | Historian reply mailbox |
| trn_cursor | USER | ANALOG | Trend cursor position |
| trn_endtime | USER | LONGANA | Trend end time (most recent) |
| trn_grid | USER | DIGITAL | Trending grid on/off |
| trn_leg_divs | USER | ANALOG | Trend chart legend divisions |

- **FACTORYLINK UTILITIES**
- *FLNEW*
-
-

Table 109-1

| Tagname | Domain | Type | Description |
|----------------|--------|---------|----------------------------------|
| trn_leg_max | USER | ANALOG | Trend chart legend max |
| trn_leg_tics | USER | ANALOG | Trend chart legend tics |
| trn_mode | USER | DIGITAL | Trend mode toggle |
| trn_pan | USER | ANALOG | Trend Pan value |
| trn_pen | USER | DIGITAL | Trend chart pen source toggle |
| trn_pen1 | USER | ANALOG | Trend cursor output value |
| trn_pen_column | USER | MESSAGE | Trend pen database column source |
| trn_starttime | USER | LONGANA | Trend Start time (oldest data) |
| trn_zoom | USER | ANALOG | Trend zoom factor |

If you restore the FactoryLink blank application (FLBLANK), the following pre-configured tags are available:

Table 109-2

| Tagname | Domain | Type | Description |
|----------------------|--------|---------|--|
| AB_EXCEPTION_DISABLE | SHARED | DIGITAL | Allen-Bradley exception write disable |
| AB_EXCEPTION_TRIGGER | SHARED | DIGITAL | Allen-Bradley exception write trigger |
| AB_LPORT0_MSG | SHARED | MESSAGE | Allen-Bradley logical port 0 error message |
| AB_READ_COMPLETE | SHARED | DIGITAL | Allen-Bradley block read complete |
| AB_READ_DISABLE | SHARED | DIGITAL | Allen-Bradley block read disable |
| AB_READ_STATE | SHARED | DIGITAL | Allen-Bradley block read state |
| AB_READ_TRIGGER | SHARED | DIGITAL | Allen-Bradley block read trigger |
| AB_STATION0_STATUS | SHARED | ANALOG | Allen-Bradley logical station 0 status |
| AB_WRITE_COMPLETE | SHARED | DIGITAL | Allen-Bradley block write complete |
| AB_WRITE_DISABLE | SHARED | DIGITAL | Allen-Bradley block write disable |
| AB_WRITE_STATE | SHARED | DIGITAL | Allen-Bradley block write state |
| AB_WRITE_TRIGGER | SHARED | DIGITAL | Allen-Bradley block write trigger |
| ALLOG_ACTIVE_COUNT | SHARED | ANALOG | Alarm Logger active alarm count |
| ALLOG_AUDIBLE_COUNT | SHARED | ANALOG | Alarm Logger audible alarm count |
| ALLOG_HIST_MBX | SHARED | MAILBOX | Alarm Logger historian mailbox |
| ALLOG_PRINT_TRIGGER | SHARED | DIGITAL | Alarm Logger print active alarms trigger |
| ALLOG_UNACK_COUNT | SHARED | ANALOG | Alarm Logger unacknowledged alarm count |
| ALVIEW_AREA | USER | MESSAGE | Alarm Viewer area filter |
| ALVIEW_BANNER | USER | MESSAGE | Alarm Viewer banner message |
| ALVIEW_BANNER_ACK | USER | DIGITAL | Alarm Viewer banner acknowledge |
| ALVIEW_BANNER_BG | USER | ANALOG | Alarm Viewer banner background attribute |
| ALVIEW_BANNER_BL | USER | ANALOG | Alarm Viewer banner blink attribute |
| ALVIEW_BANNER_FG | USER | ANALOG | Alarm Viewer banner foreground attribute |

- **FACTORYLINK UTILITIES**
- *FLNEW*
-
-

Table 109-2

| Tagname | Domain | Type | Description |
|------------------------|--------|---------|---|
| ALVIEW_GROUP | USER | MESSAGE | Alarm Viewer group filter |
| ALVIEW_GROUP_ACK | USER | DIGITAL | Alarm Viewer current group acknowledge |
| ALVIEW_LOGBOOK | USER | MESSAGE | Alarm Viewer logbook entry |
| ALVIEW_LOGBOOK_TRIGGER | USER | DIGITAL | Alarm Viewer logbook archive trigger |
| ALVIEW_ROW_SELECT | USER | ANALOG | Alarm Viewer selected row number |
| ALVIEW_SCROLL | USER | ANALOG | Alarm Viewer scroll value |
| ALVIEW_SELECT_ACK | USER | DIGITAL | Alarm Viewer selection acknowledge |
| ALVIEW_SORT | USER | MESSAGE | Alarm Viewer sort method |
| ALVIEW_TEXT | USER | MESSAGE | Alarm Viewer text |
| ALVIEW_TEXT_BG | USER | ANALOG | Alarm Viewer text background attribute |
| ALVIEW_TEXT_BL | USER | ANALOG | Alarm Viewer text blink attribute |
| ALVIEW_TEXT_FG | USER | ANALOG | Alarm Viewer text foreground attribute |
| APPLICATION_DRW | USER | MESSAGE | Application window current loaded drawing |
| A_DAY | SHARED | ANALOG | Day of the month |
| A_DOW | SHARED | ANALOG | Day of the week |
| A_DOY | SHARED | ANALOG | Day of the year |
| A_HOUR | SHARED | ANALOG | Number of hours past midnight |
| A_MIN | SHARED | ANALOG | Number of minutes past the hour |
| A_MONTH | SHARED | ANALOG | Month of the year |
| A_SEC | SHARED | ANALOG | Number of seconds past the minute |
| A_YEAR | SHARED | ANALOG | Current year |
| BROWSEHISTMBX | SHARED | MAILBOX | Mailbox for BROWSER process |
| BROWSEHISTMBX_U | USER | MAILBOX | Mailbox for BROWSER process |
| CONNSRVACTIVE | SHARED | ANALOG | Number of brokered services that are active |

Table 109-2

| Tagname | Domain | Type | Description |
|----------------|--------|---------|---|
| CONNSRVINIT | SHARED | DIGITAL | Connection server initialization flare |
| CONNSRVMBX | SHARED | MAILBOX | Connection server receive mailbox |
| CONNSRVTOTAL | SHARED | ANALOG | Total number of services being brokered |
| DALOGACKMBX | SHARED | MAILBOX | Distributed AL_LOG mailbox for acknowledged alarms |
| DALOGRCVMBX | SHARED | MAILBOX | Distributed AL_LOG receive mailbox for Historian communications |
| DALOGRCVMBX_U | USER | MAILBOX | Distributed AL_LOG receive mailbox for Historian communications |
| DALOGVIEWMBX | SHARED | MAILBOX | Distributed AL_VIEW mailbox for alarm communications |
| DALOGVIEWMBX_U | USER | MAILBOX | Distributed AL_VIEW mailbox for alarm communications |
| DATASRVMBX | USER | MAILBOX | Data server receive mailbox |
| DATE | SHARED | MESSAGE | Date (day MM/DD/YYYY) |
| DATE0 | SHARED | MESSAGE | Date (day MM/DD/YYYY) |
| DATETIME | SHARED | MESSAGE | Date and Time (day MM/DD/YYYY HH:MM:SS) |
| DB4HISTMBX | SHARED | MAILBOX | dBASEIV Historian mailbox |
| DBLOGHISTMBX | SHARED | MAILBOX | Database Logger receive mailbox for Historian communications |
| DBLOGHISTMBX_U | USER | MAILBOX | Database Logger receive mailbox for Historian communication |
| DPLOGHISTMBX | SHARED | MAILBOX | Data Point Logger receive mailbox for Historian communications |
| DYNLOGCOMMAND | SHARED | MESSAGE | Dynamic Logging command message tag for Data Point Logger |
| DYNLOGFILE | SHARED | MESSAGE | Dynamic Logging save point file for Data Point Logger |

- **FACTORYLINK UTILITIES**
- *FLNEW*
-
-

Table 109-2

| Tagname | Domain | Type | Description |
|------------------------|--------|---------|--|
| DYNLOGFILEREAD | SHARED | DIGITAL | Read trigger to load a data point save file for Data Point Logger |
| DYNLOGFILEWRITE | SHARED | DIGITAL | Write trigger to generate a Data Point Save File for Data Point Logger |
| DYNLOGSTATUS | SHARED | ANALOG | Dynamic Logging status tag for Data Point Logger |
| FLAPP_S | SHARED | MESSAGE | Application directory |
| FLAPP_U | USER | MESSAGE | Application directory |
| FLDOMAIN_S | SHARED | MESSAGE | Domain name |
| FLDOMAIN_U | USER | MESSAGE | Domain name |
| FLINK_S | SHARED | MESSAGE | Directory for application programs |
| FLINK_U | USER | MESSAGE | Directory for application programs |
| FLLANSIG | SHARED | DIGITAL | FLLANSND system interval trigger |
| FLLANSNDBOX | SHARED | MAILBOX | Mailbox for FLLANSND process |
| FLNAME_S | SHARED | MESSAGE | Application name |
| FLNAME_U | USER | MESSAGE | Application name |
| FLOPERATOR_S | SHARED | MESSAGE | Application operator |
| FLOPERATOR_U | USER | MESSAGE | Application operator |
| FLSECEVENTUSER_S | SHARED | MESSAGE | User name for application security event |
| FLSECEVENTUSER_U | USER | MESSAGE | User name for application security event |
| FLSECEVENT_S | SHARED | MESSAGE | Application security event |
| FLSECEVENT_U | USER | MESSAGE | Application security event |
| FLUSER_S | SHARED | MESSAGE | Application user |
| FLUSER_U | USER | MESSAGE | Application user |
| GENE_EXCEPTION_DISABLE | SHARED | DIGITAL | General Electric exception write disable |
| GENE_EXCEPTION_TRIGGER | SHARED | DIGITAL | General Electric exception write trigger |

Table 109-2

| Tagname | Domain | Type | Description |
|----------------------------|--------|---------|---|
| GENE_LPORT0_MSG | SHARED | MESSAGE | General Electric logical port 0 error message |
| GENE_READ_COMPLETE | SHARED | DIGITAL | General Electric block read complete |
| GENE_READ_DISABLE | SHARED | DIGITAL | General Electric block read disable |
| GENE_READ_STATE | SHARED | DIGITAL | General Electric block read state |
| GENE_READ_TRIGGER | SHARED | DIGITAL | General Electric block read trigger |
| GENE_STATION0_STATUS | SHARED | ANALOG | General Electric logical station 0 status |
| GENE_WRITE_COMPLETE | SHARED | DIGITAL | General Electric block write complete |
| GENE_WRITE_DISABLE | SHARED | DIGITAL | General Electric block write disable |
| GENE_WRITE_STATE | SHARED | DIGITAL | General Electric block write state |
| GENE_WRITE_TRIGGER | SHARED | DIGITAL | General Electric block write trigger |
| GRAPHCONNTYPE | USER | ANALOG | Graph connection and security type |
| GRAPHMBX | SHARED | MAILBOX | Graphics input mailbox |
| GRAPHMBX_U | USER | MAILBOX | Graphics input mailbox |
| H1MP_EXCEPTION_DISABLE | SHARED | DIGITAL | Siemens Sinec H1 exception write disable |
| H1MP_EXCEPTION_TRIGGER | SHARED | DIGITAL | Siemens Sinec H1 exception write trigger |
| H1MP_LPORT0_MSG | SHARED | MESSAGE | Siemens Sinec H1 logical port 0 error message |
| H1MP_READ_COMPLETE | SHARED | DIGITAL | Siemens Sinec H1 block read complete |
| H1MP_READ_DISABLE | SHARED | DIGITAL | Siemens Sinec H1 block read disable |
| H1MP_READ_STATE | SHARED | DIGITAL | Siemens Sinec H1 block read state |
| H1MP_READ_TRIGGER | SHARED | DIGITAL | Siemens Sinec H1 block read trigger |
| H1MP_STATION0_READ_STATUS | SHARED | DIGITAL | Siemens Sinec H1 solicited read status |
| H1MP_STATION0_STATUS | SHARED | ANALOG | Siemens Sinec H1 logical station 0 status |
| H1MP_STATION0_UNSol_STATUS | SHARED | DIGITAL | Siemens Sinec H1 unsolicited read status |
| H1MP_STATION0_WRITE_STATUS | SHARED | DIGITAL | Siemens Sinec H1 solicited write status |

- **FACTORYLINK UTILITIES**
- *FLNEW*
-
-

Table 109-2

| Tagname | Domain | Type | Description |
|-------------------------|--------|---------|--|
| H1MP_WRITE_COMPLETE | SHARED | DIGITAL | Siemens Sinec H1 block write complete |
| H1MP_WRITE_DISABLE | SHARED | DIGITAL | Siemens Sinec H1 block write disable |
| H1MP_WRITE_STATE | SHARED | DIGITAL | Siemens Sinec H1 block write state |
| H1MP_WRITE_TRIGGER | SHARED | DIGITAL | Siemens Sinec H1 block write trigger |
| HELP_COMPLETE | USER | DIGITAL | Help file read or position complete |
| HELP_SCROLL | USER | ANALOG | Help file position used for scrolling |
| HELP_STATUS | USER | ANALOG | Help file status |
| HELP_TEXT | USER | MESSAGE | Help file output text |
| HELP_TRIGGER | USER | DIGITAL | Help file read trigger |
| KTDTL_EXCEPTION_DISABLE | SHARED | DIGITAL | Allen-Bradley KTDTL exception write disable |
| KTDTL_EXCEPTION_TRIGGER | SHARED | DIGITAL | Allen-Bradley KTDTL exception write trigger |
| KTDTL_LPORT0_MSG | SHARED | MESSAGE | Allen-Bradley KTDTL logical port 0 error message |
| KTDTL_READ_COMPLETE | SHARED | DIGITAL | Allen-Bradley KTDTL block read complete |
| KTDTL_READ_DISABLE | SHARED | DIGITAL | Allen-Bradley KTDTL block read disable |
| KTDTL_READ_STATE | SHARED | DIGITAL | Allen-Bradley KTDTL block read state |
| KTDTL_READ_TRIGGER | SHARED | DIGITAL | Allen-Bradley KTDTL block read trigger |
| KTDTL_STATION0_STATUS | SHARED | LONGANA | Allen-Bradley KTDTL logical station 0 status |
| KTDTL_WRITE_COMPLETE | SHARED | DIGITAL | Allen-Bradley KTDTL block write complete |
| KTDTL_WRITE_DISABLE | SHARED | DIGITAL | Allen-Bradley KTDTL block write disable |
| KTDTL_WRITE_STATE | SHARED | DIGITAL | Allen-Bradley KTDTL block write state |
| KTDTL_WRITE_TRIGGER | SHARED | DIGITAL | Allen-Bradley KTDTL block write trigger |
| MBUS_EXCEPTION_DISABLE | SHARED | DIGITAL | Modicon Modbus exception write disable |
| MBUS_EXCEPTION_TRIGGER | SHARED | DIGITAL | Modicon Modbus exception write trigger |

Table 109-2

| Tagname | Domain | Type | Description |
|------------------------|--------|---------|--|
| MBUS_LPORT0_MSG | SHARED | MESSAGE | Modicon Modbus logical port 0 error message |
| MBUS_READ_COMPLETE | SHARED | DIGITAL | Modicon Modbus block read complete |
| MBUS_READ_DISABLE | SHARED | DIGITAL | Modicon Modbus block read disable |
| MBUS_READ_STATE | SHARED | DIGITAL | Modicon Modbus block read state |
| MBUS_READ_TRIGGER | SHARED | DIGITAL | Modicon Modbus block read trigger |
| MBUS_STATION0_STATUS | SHARED | ANALOG | Modicon Modbus logical station 0 status |
| MBUS_WRITE_COMPLETE | SHARED | DIGITAL | Modicon Modbus block write complete |
| MBUS_WRITE_DISABLE | SHARED | DIGITAL | Modicon Modbus block write disable |
| MBUS_WRITE_STATE | SHARED | DIGITAL | Modicon Modbus block write state |
| MBUS_WRITE_TRIGGER | SHARED | DIGITAL | Modicon Modbus block write trigger |
| MODP_EXCEPTION_DISABLE | SHARED | DIGITAL | Modicon Modbus Plus exception write disable |
| MODP_EXCEPTION_TRIGGER | SHARED | DIGITAL | Modicon Modbus Plus exception write trigger |
| MODP_LPORT0_MSG | SHARED | MESSAGE | Modicon Modbus Plus logical port 0 error message |
| MODP_LS0_FAIL_DIG | SHARED | DIGITAL | Modicon Modbus Plus logical station failure |
| MODP_LS0_FAIL_MSG | SHARED | MESSAGE | Modicon Modbus Plus communication failure |
| MODP_READ_COMPLETE | SHARED | DIGITAL | Modicon Modbus Plus block read complete |
| MODP_READ_DISABLE | SHARED | DIGITAL | Modicon Modbus Plus block read disable |
| MODP_READ_STATE | SHARED | DIGITAL | Modicon Modbus Plus block read state |
| MODP_READ_TRIGGER | SHARED | DIGITAL | Modicon Modbus Plus block read trigger |
| MODP_STATION0_STATUS | SHARED | ANALOG | Modicon Modbus Plus logical station 0 status |
| MODP_WRITE_COMPLETE | SHARED | DIGITAL | Modicon Modbus Plus block write complete |
| MODP_WRITE_DISABLE | SHARED | DIGITAL | Modicon Modbus Plus block write disable |
| MODP_WRITE_STATE | SHARED | DIGITAL | Modicon Modbus Plus block write state |
| MODP_WRITE_TRIGGER | SHARED | DIGITAL | Modicon Modbus Plus block write trigger |

- **FACTORYLINK UTILITIES**
- *FLNEW*
-
-

Table 109-2

| Tagname | Domain | Type | Description |
|------------------------|--------|---------|---|
| NDTL_EXCEPTION_DISABLE | SHARED | DIGITAL | Allen-Bradley NetDTL exception write disable |
| NDTL_EXCEPTION_TRIGGER | SHARED | DIGITAL | Allen-Bradley NetDTL exception write trigger |
| NDTL_LPORT0_MSG | SHARED | MESSAGE | Allen-Bradley NetDTL logical port 0 error message |
| NDTL_READ_COMPLETE | SHARED | DIGITAL | Allen-Bradley NetDTL block read complete |
| NDTL_READ_DISABLE | SHARED | DIGITAL | Allen-Bradley NetDTL block read disable |
| NDTL_READ_STATE | SHARED | DIGITAL | Allen-Bradley NetDTL block read state |
| NDTL_READ_TRIGGER | SHARED | DIGITAL | Allen-Bradley NetDTL block read trigger |
| NDTL_STATION0_STATUS | SHARED | LONGANA | Allen-Bradley NetDTL logical station 0 status |
| NDTL_WRITE_COMPLETE | SHARED | DIGITAL | Allen-Bradley NetDTL block write complete |
| NDTL_WRITE_DISABLE | SHARED | DIGITAL | Allen-Bradley NetDTL block write disable |
| NDTL_WRITE_STATE | SHARED | DIGITAL | Allen-Bradley NetDTL block write state |
| NDTL_WRITE_TRIGGER | SHARED | DIGITAL | Allen-Bradley NetDTL block write trigger |
| OMRN_EXCEPTION_DISABLE | SHARED | DIGITAL | Omron exception write disable |
| OMRN_EXCEPTION_TRIGGER | SHARED | DIGITAL | Omron exception write trigger |
| OMRN_READ_COMPLETE | SHARED | DIGITAL | Omron block read complete |
| OMRN_READ_DISABLE | SHARED | DIGITAL | Omron block read disable |
| OMRN_READ_STATE | SHARED | DIGITAL | Omron block read state |
| OMRN_READ_TRIGGER | SHARED | DIGITAL | Omron block read trigger |
| OMRN_STATION0_STATUS | SHARED | ANALOG | Omron logical station 0 status |
| OMRN_WRITE_COMPLETE | SHARED | DIGITAL | Omron block write complete |
| OMRN_WRITE_DISABLE | SHARED | DIGITAL | Omron block write disable |
| OMRN_WRITE_STATE | SHARED | DIGITAL | Omron block write state |

Table 109-2

| Tagname | Domain | Type | Description |
|------------------------|--------|---------|--|
| OMRN_WRITE_TRIGGER | SHARED | DIGITAL | Omron block write trigger |
| RTMARG | SHARED | ANALOG | Run-Time Manager argument |
| RTMARG_U | USER | ANALOG | Run-Time Manager argument |
| RTMCMD | SHARED | ANALOG | Run-Time Manager system Command |
| RTMCMD_U | USER | ANALOG | Run-Time Manager system Command |
| RTMPWD | SHARED | MESSAGE | Run-Time Manger password |
| RTMPWD_U | USER | MESSAGE | Run-Time Manger password |
| SECDAY | SHARED | LONGANA | Number of seconds past midnight |
| SECTIME | SHARED | LONGANA | Number of seconds since \ |
| SECURITYMBX | SHARED | MAILBOX | Mailbox tag to log information collected by security |
| SECYEAR | SHARED | LONGANA | Number of seconds past January 1\ |
| SHUTDOWN | SHARED | DIGITAL | Run-Time Manager shutdown flag |
| SHUTDOWN_U | USER | DIGITAL | Run-Time Manager shutdown flag |
| SIEM_EXCEPTION_DISABLE | SHARED | DIGITAL | Siemens CP525 exception write disable |
| SIEM_EXCEPTION_TRIGGER | SHARED | DIGITAL | Siemens CP525 exception write trigger |
| SIEM_LPORT0_MSG | SHARED | MESSAGE | Siemens CP525 logical port 0 error message |
| SIEM_READ_COMPLETE | SHARED | DIGITAL | Siemens CP525 block read complete |
| SIEM_READ_DISABLE | SHARED | DIGITAL | Siemens CP525 block read disable |
| SIEM_READ_STATE | SHARED | DIGITAL | Siemens CP525 block read state |
| SIEM_READ_TRIGGER | SHARED | DIGITAL | Siemens CP525 block read trigger |
| SIEM_STATION0_STATUS | SHARED | ANALOG | Siemens CP525 logical station 0 status |
| SIEM_WRITE_COMPLETE | SHARED | DIGITAL | Siemens CP525 block write complete |
| SIEM_WRITE_DISABLE | SHARED | DIGITAL | Siemens CP525 block write disable |
| SIEM_WRITE_STATE | SHARED | DIGITAL | Siemens CP525 block write state |

- **FACTORYLINK UTILITIES**
- *FLNEW*
-
-

Table 109-2

| Tagname | Domain | Type | Description |
|------------------------|--------|---------|---|
| SIEM_WRITE_TRIGGER | SHARED | DIGITAL | Siemens CP525 block write trigger |
| SPCDATAMBX_S | SHARED | MAILBOX | PowerSPC Data task receive mailbox |
| SPCDATATRIG_S | SHARED | DIGITAL | PowerSPC Data task initialization indicator |
| SPCGMBX | SHARED | MAILBOX | SPC Graphics input mailbox |
| SPCGMBX_U | USER | MAILBOX | SPC Graphics input mailbox |
| SPCGRPHMBX_S | SHARED | MAILBOX | PowerSPC Graphics task receive mailbox |
| SPCGRPHMBX_U | USER | MAILBOX | PowerSPC Graphics task receive mailbox |
| SPCRCVMBX | SHARED | MAILBOX | SPC receive mailbox for database access |
| SPCRCVMBX_U | USER | MAILBOX | SPC receive mailbox for database access |
| SPOOLREQ | SHARED | MESSAGE | Print Spooler request |
| SPOOLRPLY | SHARED | MESSAGE | Print Spooler reply |
| SPRGMBX | SHARED | MAILBOX | SPR graphics input mailbox |
| SPRGMBX_U | USER | MAILBOX | SPR graphics input mailbox |
| SPRRCVMBX | SHARED | MAILBOX | SPR receive mailbox for database access |
| SPRRCVMBX_U | USER | MAILBOX | SPR receive mailbox for database access |
| SPVRCVMBX | SHARED | MAILBOX | SPC View mailbox for database access |
| SPVRCVMBX_U | USER | MAILBOX | SPC View mailbox for database access |
| SQRD_EXCEPTION_DISABLE | SHARED | DIGITAL | Square D exception write disable |
| SQRD_EXCEPTION_TRIGGER | SHARED | DIGITAL | Square D exception write trigger |
| SQRD_LPORT0_MSG | SHARED | MESSAGE | Square D logical port 0 error message |
| SQRD_READ_COMPLETE | SHARED | DIGITAL | Square D block read complete |
| SQRD_READ_DISABLE | SHARED | DIGITAL | Square D block read disable |
| SQRD_READ_STATE | SHARED | DIGITAL | Square D block read state |
| SQRD_READ_TRIGGER | SHARED | DIGITAL | Square D block read trigger |

Table 109-2

| Tagname | Domain | Type | Description |
|----------------------|--------|---------|--|
| SQRD_STATION0_STATUS | SHARED | ANALOG | Square D logical station 0 status |
| SQRD_WRITE_COMPLETE | SHARED | DIGITAL | Square D block write complete |
| SQRD_WRITE_DISABLE | SHARED | DIGITAL | Square D block write disable |
| SQRD_WRITE_STATE | SHARED | DIGITAL | Square D block write state |
| SQRD_WRITE_TRIGGER | SHARED | DIGITAL | Square D block write trigger |
| STARTUP | SHARED | DIGITAL | Run-Time Manager startup flag |
| STARTUP_U | USER | DIGITAL | Run-Time Manager startup flag |
| TASKDESC_S | SHARED | MESSAGE | Shared Task Description |
| TASKDESC_U | USER | MESSAGE | User Task Description |
| TASKDSTATUS_S | SHARED | MESSAGE | Shared Task Status Word |
| TASKDSTATUS_U | USER | MESSAGE | User Task Display Status Message |
| TASKMESSAGE_S | SHARED | MESSAGE | Shared Task Display Message |
| TASKMESSAGE_U | USER | MESSAGE | User Task Message |
| TASKNAME_S | SHARED | MESSAGE | Shared Task Name |
| TASKNAME_U | USER | MESSAGE | User Task Name |
| TASKSTART_S | SHARED | DIGITAL | Shared Task Start Trigger |
| TASKSTART_U | USER | DIGITAL | User Task Start Trigger |
| TASKSTATUS_S | SHARED | ANALOG | Shared Task Status Value |
| TASKSTATUS_U | USER | ANALOG | User Task Status Value |
| TE_EXCEPTION_DISABLE | SHARED | DIGITAL | Telemecanique exception write disable |
| TE_EXCEPTION_TRIGGER | SHARED | DIGITAL | Telemecanique exception write trigger |
| TE_LPORT0_MSG | SHARED | MESSAGE | Telemecanique logical port 0 error message |
| TE_READ_COMPLETE | SHARED | DIGITAL | Telemecanique block read complete |
| TE_READ_DISABLE | SHARED | DIGITAL | Telemecanique block read disable |

- **FACTORYLINK UTILITIES**
- *FLNEW*
-
-

Table 109-2

| Tagname | Domain | Type | Description |
|----------------------|--------|---------|--|
| TE_READ_STATE | SHARED | DIGITAL | Telemecanique block read state |
| TE_READ_TRIGGER | SHARED | DIGITAL | Telemecanique block read trigger |
| TE_STATION0_STATUS | SHARED | ANALOG | Telemecanique logical station 0 status |
| TE_WRITE_COMPLETE | SHARED | DIGITAL | Telemecanique block write complete |
| TE_WRITE_DISABLE | SHARED | DIGITAL | Telemecanique block write disable |
| TE_WRITE_STATE | SHARED | DIGITAL | Telemecanique block write state |
| TE_WRITE_TRIGGER | SHARED | DIGITAL | Telemecanique block write trigger |
| TIME | SHARED | MESSAGE | Time (HH:MM:SS) |
| TIME0 | SHARED | MESSAGE | Time (HH:MM:SS) |
| TI_EXCEPTION_DISABLE | SHARED | DIGITAL | Texas Instruments exception write disable |
| TI_EXCEPTION_TRIGGER | SHARED | DIGITAL | Texas Instruments exception write trigger |
| TI_LPORT0_MSG | SHARED | MESSAGE | Texas Instruments logical port 0 error message |
| TI_READ_COMPLETE | SHARED | DIGITAL | Texas Instruments block read complete |
| TI_READ_DISABLE | SHARED | DIGITAL | Texas Instruments block read disable |
| TI_READ_STATE | SHARED | DIGITAL | Texas Instruments block read state |
| TI_READ_TRIGGER | SHARED | DIGITAL | Texas Instruments block read trigger |
| TI_STATION0_STATUS | SHARED | ANALOG | Texas Instruments logical station 0 status |
| TI_WRITE_COMPLETE | SHARED | DIGITAL | Texas Instruments block write complete |
| TI_WRITE_DISABLE | SHARED | DIGITAL | Texas Instruments block write disable |
| TI_WRITE_STATE | SHARED | DIGITAL | Texas Instruments block write state |
| TI_WRITE_TRIGGER | SHARED | DIGITAL | Texas Instruments block write trigger |
| TOPWINDOW_U | USER | MESSAGE | Current top window name |
| TRENDBHISTMBX | SHARED | MAILBOX | TREND receive mailbox for historian communications |

Table 109-2

| Tagname | Domain | Type | Description |
|-------------------------|--------|---------|---|
| TRENDHISTMBX_U | USER | MAILBOX | TREND receive mailbox for historian communication |
| TRENDBOX | SHARED | MAILBOX | Trend input mailbox |
| TRENDBOX_U | USER | MAILBOX | Trend input mailbox |
| VBLOGDISABLE_S | SHARED | DIGITAL | PowerVB debug logging disable |
| VBLOGDISABLE_U | USER | DIGITAL | PowerVB debug logging disable |
| YYMMDD | SHARED | MESSAGE | Date (YYMMDD) |
| s1process_button_action | USER | DIGITAL | Style 1 process button tag used for VB events |
| sec1 | SHARED | DIGITAL | One second interval timer trigger |

- **FACTORYLINK UTILITIES**
- *FLREST*
-
-

FLREST

Use the FLREST utility to restore saved FactoryLink application files. FLREST overwrites existing files. Never restore to the root directory, the FLINK directory, or any other directory that contains working files. The safest option is to have an independent directory for restores, preferably on a separate disk or partition.

FLREST restores FactoryLink applications by one of two methods:

- Platform-specific
- Multi-platform

Platform-specific restore

The platform-specific method restores an application from another computer running on the same operating system. Use the platform-specific restore

- to restore platform-specific files.
- to restore backed up applications.

This restores the entire application.

Multiplatform restore

The multiplatform method transfers an application from a FactoryLink system running on one operating system to a FactoryLink system running on a different operating system. The multiplatform method expands a FactoryLink application from a single image file of the application. Use the multiplatform restore when transporting applications across platforms.

If the application you are restoring was built under a FactoryLink version prior to 4.3, your current operating system platform may not allow file names, directory names, and graphic screens to be longer than 8 characters, not including the file name extension. Starting with 4.3, file names, directory names, and graphic screens cannot exceed 8 characters. If they do, the application will not restore on FactoryLink 4.3 or higher.

Only restore a multiplatform save to the same FactoryLink version from which it was saved because configuration data may be lost.

Restoring an application

Perform the following steps to restore an application from a platform-specific or multiplatform backup.

1 Insert the source diskette into the drive.

2 Start the FLREST utility. The FactoryLink Application Restore dialog is displayed.

3 Specify the following information in the dialog.

Source Specifies the application drive and path. The default is a:\.

Destination Specifies the destination drive and path. The default is c:\flapp\
If you do not specify a filename in the destination path, a
multiplatform FLREST will fail.

4 Choose the type of restore to perform. This can be one of the following:

Local Backup Performs a platform-specific restore.

Local Multiplatform
Restore File Performs a multiplatform save to a local drive.

Network
Multiplatform Save
File Performs a multiplatform restore from a remote node to the
current node. Choose this option if you want to restore the
application from a node running FactoryLink under another
operating system.

This option uses the TCP/IP executable `rcp` to copy
multiplatform restores between nodes on the network. This
requires the source node to be configured as a network server and
the destination node hostname to exist within the source
HOST.EQU file.

5 FLREST spans diskettes for only the Local Backup format. Local Multiplatform
restores do not span diskettes.

Choose OK or Cancel when you complete this panel.

OK Initiates the restore.

CANCEL Exits without restoring the application.

- **FACTORYLINK UTILITIES**

- *FLREST*

-
-

The following files are restored by both platform-specific and multiplatform FLREST operations.

- Configuration tables (*.CDB, .MDB)
- Graphics files (DRW/*.G, *.GC, *.GS, *.GP, *.GPC, *.GPS, *.PL, *.PLS, *.PLC)
- Report Format files (RPT/*.FMT)
- Math & Logic source field (PROCS/ *.PRG, *.INC)
- CML source files (CML/*.C, *.H, *MAK)
- Network configuration files (NET/*.*)

The following files are restored by platform-specific FLREST operations.

- CML binary files (CML/*.OBJ, *.EXE)
- Log files (LOG/*.*)
- Recipe files (RCP/*.*)

- 6 When all files have been copied to the appropriate drive/directory, the following message is displayed.

FactoryLink Application Restore Completed

- 7 Restore CML binary, Log, or Recipe files with a multiplatform FLREST by copying them manually from the backup diskette.

FLSETLNG

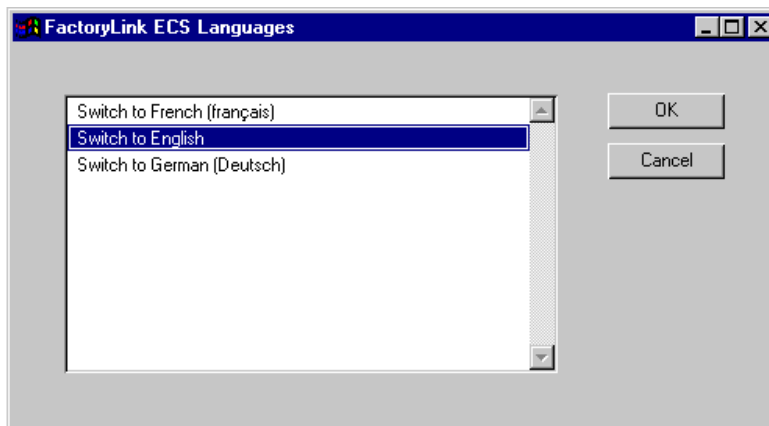
The FLSETLNG utility allows the user to specify the language FactoryLink Configuration Manager, Application Editor, and non-operating system specific Runtime modules run in based on the languages installed on the system. FLSETLNG can be run stand-alone or by double clicking on the executable file.

Caution: You will receive an error message if you try to run FLSETLNG while another FactoryLink task or configuration is running.

The executable file is located in the {FLINK}\BIN\ directory. To run FLSETLNG in stand-alone mode, enter the following command at the system prompt:

```
flsetlng
```

The FactoryLink 6.5.0 Languages dialog displays the available languages installed on your system that you can choose to run your applications.



Specify the language you want and click on OK. A confirmation box displays if the language has been set to your selection.

If you are using multiple languages in development, be aware that language-specific characters will not translate after performing a language switch.

For example, a change in the security password in one language will not work in another if you are using language-specific characters (accent mark).

If it is necessary to have unique national characters working, enter the 1252 code page compatible ASCII code via the ALT (keyboard numbers).

- **FACTORYLINK UTILITIES**

- *FLTEST and FLDEMO*

-
-

FLTEST AND FLDEMO

After installing FactoryLink, run the test and demo applications, FLTEST and FLDEMO, to test the FactoryLink installation.

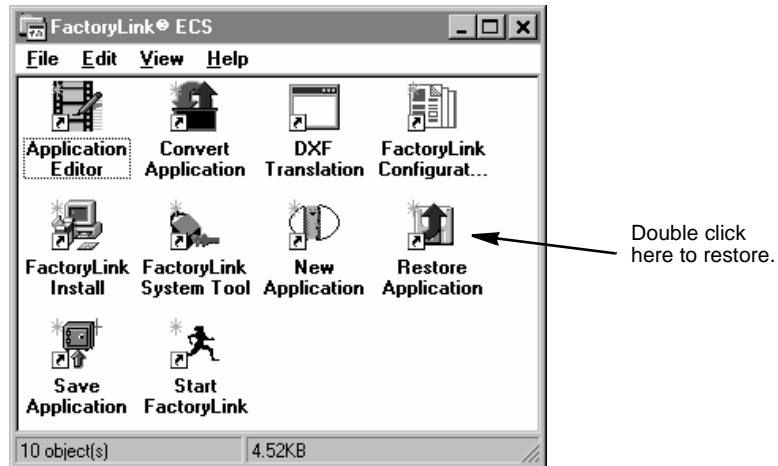
- FLTEST is a test application for testing the installation of FactoryLink. It also provides examples of common tasks.
- FLDEMO provides you with application development ideas. You can watch its execution by starting the FactoryLink Run-Time Manager only if restored FLAPP is set.

This section describes how to restore the FLTEST or FLDEMO application on each operating system supported by FactoryLink. The process varies for each operating system.

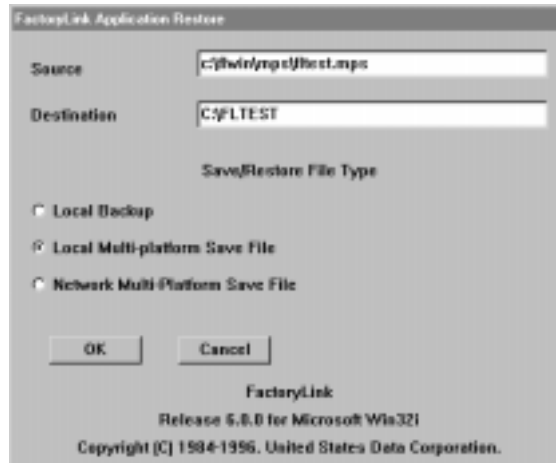
On Windows NT and Windows 95 Operating Systems

Perform the following steps to restore FLTEST or FLDEMO on a Windows NT or Windows 95 operating system.

- 1 Choose the Restore Application icon from the FactoryLink group window.



- 2 The Application Restore dialog is displayed.



- 3 Provide the following information.

Source Enter the full path name of the application's source directory.

If you are restoring FLTEST, enter

`c:\flink\mps\fltest.mps`

If you are restoring FLDEMO, enter

`c:\flink\mps\fldemo.mps`

Destination Enter the full path name of the application's destination directory.

- 4 Choose Local Multiplatform Save File as the type of save file you want to restore.
- 5 Choose OK. FLREST begins restoring the files.
- 6 Create a program group for the FLTEST application using Additional Install Functions from the Windows Program Manager.

These applications are configured to use the dBASE IV Historian. If you do not use this Historian, configure the application to use the desired Historian before you run any activities accessing a relational database.

This application uses most of the FactoryLink tasks you have purchased. Some screens may do nothing if you have not purchased or installed the task that provides the information to that screen.

- **FACTORYLINK UTILITIES**

- *FLTEST and FLDEMO*

-
-

On OS/2 Operating Systems

This section describes how to restore the FLTEST or FLDEMO application on an OS/2 operating system from the Program Manager and a command line.

From the Desktop Manager

- 1 Choose FactoryLink OS/2 program group from the Desktop Manager menu to display the applications available with the FactoryLink group.

Double click
here to restore
an application.



- 2 Choose Restore Application.
- 3 Follow the onscreen instructions.

From a Command Line

- 1 Change to the MPS directory in the FLINK path at the OS/2 system prompt. For example,

```
CD \FLOS2\MPS
```

- 2 Enter the following command if you want to restore the FLTEST application.

```
FLREST FLTEST.MPS c:\FLTEST /loc
```

Enter the following command if you want to restore the FLDEMO application.

```
FLREST FLTEST.MPS c:\FLDEMO /loc
```

- 3 Provide the following information

Source Enter the full path name of the application source directory.

If you are restoring FLTEST, enter

```
c:\flink\mps\fltest.mps
```

If you are restoring FLDEMO, enter

```
c:\flink\mps\fldemo.mps
```

Destination Enter the full path name of the application destination directory.

- 4 Choose Local Multiplatform Save File as the type of save file you want to restore.
- 5 Choose OK. FLREST begins restoring the files.
- 6 Create a program group for the FLTEST application using Additional Install Functions from the Windows Program Manager.

These application are configured to use the dBASE IV Historian. If you do not use this Historian, configure the application to use the desired Historian before you run any activities opening a relational database.

This application uses most of the FactoryLink tasks you have purchased. Some screens may do nothing if you have not purchased or installed the task that provides the information to that screen.

- **FACTORYLINK UTILITIES**
- *On UNIX Operating Systems*
-
-

ON UNIX OPERATING SYSTEMS

This section describes how to restore the FLDEMO or FLTEST and applications on a UNIX operating system.

Restoring FLDEMO

- 1 Create a target directory for the test application by entering the following command at the system prompt.**
`mkdir fldemo`
- 2 Change to the newly created directory by entering the following command at the system prompt.**
`cd fldemo`
- 3 Enter the following command to restore multiplatform save file for FLDEMO.**
`flrest FLINK/mps/fldemo.mps /fldemo /loc`

Restoring FLTEST

Perform the following steps to restore FLTEST on a UNIX operating system.

- 1 Create a target directory for the test application by entering the following command at the system prompt.**
`mkdir fltest`
- 2 Change to the newly created directory by entering the following command at the system prompt.**
`cd fltest`
- 3 Enter the following command to restore multiplatform save file for FLTEST.**
`flrest FLINK/mps/fltest.mps /usr/flapps/fltest /loc`

MODULE DEPENDENCIES

Refer to the following chart to determine the tasks required to run the appropriate applications when installing FactoryLink.

Table 109-3

| FLDEMO | | FLTEST | |
|--------------------------|-----------------------------------|--------------------------|-----------------------------------|
| Alarm Supervisor | Report Generator | Alarm Supervisor | Persistence |
| Configuration Manager | Statistical Process Control (SPC) | Configuration Manager | Batch Recipe |
| Run-Time Color Graphics | Statistical Process Recalculation | Run-Time Color Graphics | Report Generator |
| Database Browser | Timer | Database Browser | SPR |
| Database Logger | Counter | Database Logger | Statistical Process Control (SPC) |
| DB4 Historian | Trending | DB4 Historian | Print Spooler |
| File Manager | | File Manager | Timer |
| Interpreted Math & Logic | | Interpreted Math & Logic | Trending |
| Batch Recipe | | Counter | |

- **FACTORYLINK UTILITIES**

- *FLCONV*

-

-

FLCONV

After restoring a FactoryLink application on a new platform or a new FactoryLink version, you must convert it using the FLCONV utility. Once you convert the application, it will not run with the old FactoryLink.

Do not abort the convert after it has started as this can corrupt the application.

Perform the following steps to convert an application.

- 1 Back-up the application using a platform-specific FLSAVE if you have not already done so. Refer to FLSAVE in this section for more information.
- 2 Start the FLCONV utility. A dialog is displayed containing the drive and path of the application to convert. Verify the drive and path are correct.
- 3 Choose one of the following:
 - OK Converts the application.
 - Cancel Cancels the conversion.

The application files are converted.

FLSAVE

Use the FLSAVE utility to save FactoryLink application files. FLSAVE overwrites existing files. Never save to the root directory, the FLINK directory, or any other directory that contains working files. The safest option is to have an independent directory for saves, preferably on a separate disk or partition.

FLSAVE saves FactoryLink applications by one of two methods:

- Platform-specific
- Multi-platform

Platform-specific save

The platform-specific method creates a copy of an application that can be restored to other computers running on the same operating system. Use the platform-specific save to:

- save platform-specific files.
- back up applications.

This creates a complete archive of the entire application.

Multiplatform save

The multiplatform method transfers an application from a FactoryLink system running on one operating system to a FactoryLink system running on a different operating system. The multiplatform method condenses a FactoryLink application into a single image file of the application. Use the multiplatform save when transporting applications across platforms.

If you are converting an application built under a FactoryLink version prior to 4.3, your operating system platform may have allowed file names, directory names, and graphic screens to be longer than 8 characters, not including the file name extension. Starting with 4.3, file names, directory names, and graphic screens cannot exceed 8 characters. If they do, the application will not restore on FactoryLink 4.3 or higher.

Never restore a multiplatform save to a FactoryLink version other than the version with which it was saved because configuration data may be lost.

- **FACTORYLINK UTILITIES**
- *FLSAVE*
-
-

Saving an application

Perform the following steps to save an application for a platform-specific or multiplatform backup.

- 1 Format (initialize) the target diskette before starting FLSAVE. We recommend using a blank diskette because FLSAVE destroys any applications existing on the diskette.
- 2 Insert the target diskette into the drive.
- 3 Start the FLSAVE utility. The FactoryLink Application Save dialog is displayed.
- 4 Specify the following information in the dialog.

| | |
|-----------------------------|---|
| Source | Specifies the application drive and path. The default is c:\flapp. |
| Destination | Specifies the destination drive and path. The default is a:\. If you do not specify a filename in the destination path, a multiplatform FLSAVE will fail. |
| Save Application Data Files | Specifies whether or not to save all application data files. To use this option, click on the selection box. |

- 5 Choose the type of save to perform. This can be one of the following:

| | |
|---------------------------------|---|
| Local Backup | Performs a platform-specific save. |
| Local Multiplatform Save File | Performs a multiplatform save to a local drive. |
| Network Multiplatform Save File | Performs a multiplatform save of the current node to a remote node. Choose this option if you want to restore the application on a node running FactoryLink under another operating system. This option uses the TCP/IP executable <code>ncp</code> to copy multiplatform saves between nodes on the network. This requires the destination node be configured as a network server and the source node hostname exists within the destination HOST.EQU file. |

FLSAVE spans diskettes for only the Local Backup format. Local Multiplatform saves do not span diskettes.

- 6 Choose OK or Cancel when you complete this panel.

| | |
|--------|---------------------------------------|
| OK | Initiates the save. |
| CANCEL | Exits without saving the application. |

The following files are saved by both platform-specific and multiplatform FLSAVE operations.

- Configuration tables (*.CDB, *.MDB)
- Graphics files (DRW/*.G, *.GC, *.GS, *.GP, *.GPC, *.GPS, *.PL, *.PLS, *.PLC)
- Report Format files (RPT/*.FMT)
- Math & Logic source field (PROCS/ *.PRG, *.INC)
- CML source files (CML/*.C, *.H, *.MAK)
- Network configuration files (NET/*.*)

The following files are saved by platform-specific FLSAVE operations.

- CML binary files (CML/*.OBJ, *.EXE)
- Log files (LOG/*.*)
- Recipe files (RCP/*.*)

- 7 When all files have been copied to the appropriate drive/directory or to diskettes, the following message is displayed.

FactoryLink Application Save Completed

- 8 Save CML binary, Log, or Recipe files with a multiplatform FLSAVE by copying them manually onto a backup diskette.

- **FACTORYLINK UTILITIES**
- *CTGEN*
-
-

CTGEN

The CTGEN utility binds or converts the tagnames specified in the database tables to tag numbers maintained by the real-time database. At run time, the task loads the CT file and builds any internal structures required to perform the job. To improve performance, tasks use the tag number from the CT instead of the tagnames in the database table(s) to access the real-time database.

CTGEN uses the CTLIST file to build CTs and rebuild all CTs whose database tables have changed. CTGEN can be run stand-alone or with a combination of parameters. To run CTGEN in stand-alone mode, enter the following command at the system prompt:

```
ctgen <Enter>
```

CDBLIST

This utility is used to debug at the database configuration level.

The command line is

```
cdblist [-d] <file.cdb> [<file.mdx>]
```

where

-d shows field names
 <file.cdb> is the database file to show contents

CTLIST

This utility is used to debug at the binary level.

The command line is

```
ctlist <file.ct>
```

DBCHK

Use the Database Checking (DBCHK) utility to

- Check databases for corrupted index files.
- Remove duplicate entries.
- Rebuild index files.

The syntax is

`dbchk [options]`

where

options controls how `dbchk` executes. If you do not specify any command line parameters, the utility only reports on potential problems for the whole application without trying to correct them. *options* can be one of the following.

- A *flapp_dir* defines the name of the application directory to check where *flapp_dir* is the full path to the directory. If you do not specify *flapp_dir*, the default FLAPP directory is used. Be sure not to use a space between the -A option and *flapp_dir*.
- T *titlefile* defines the files to check, where *titlefile* is the name of the file containing the names of the AC files to check. *titlefile* can be the name of any file you created using a text editor.
 FLINK\AC\TTITLES—path of the file that contains the names of all AC files accessible from the Configuration Manager Main Menu. Use this option if you want to check all the files.
 FLINK\AC\TTITLES—name of the file that contains the names of all AC files not directly accessible from the Main Menu, like the object and cross-reference databases. Be sure not to use a space between the -T option and *titlefile*.
- C reports potential problems but does not try to correct them. If this option is chosen, only reporting occurs even if other options are specified.
- D removes duplicate entries from the database files and generates a new index if any are found. Be careful not to specify a file that permits duplicate entries or they will be removed.
- I generates a new index for the database files.

- **FACTORYLINK UTILITIES**

- *EXPLODE*

-
-

-Vn activates verbose mode where *n* indicates the level of verbosity. This can be one of the following:

1 provides only record number messages

2 provides text messages

-W checks the entire application. This means all AC files listed in *titlefile* plus OBJECT.AC, XREF.AC, DOMAIN.AC, and TYPE.AC are checked.

ac_file.AC is the name of the AC file or list of files to check. You can specify up to 200 AC files. Each name must be on a separate line.

>filename.out redirects the output to *filename.out* where *filename.out* is the name of the file to receive the output.

EXPLODE

If a FactoryLink system file, not an application file, becomes corrupt and must be replaced on the hard drive from the installation media, use the `explode` utility to decompress individual files before replacing them. This is only necessary for Windows NT, Windows 95, and OS/2 platforms. This is necessary because the Installation and System Software media contain compressed files. The syntax is

`explode infile outfile`

where

infile is the full path of the file you want to copy from the installation disk including the name of the drive where the installation disk is mounted.

outfile is the drive and full path name of the directory where you want to write the file.

KEYINST

This is a protection utility usually run during installation; however, you can run it at any time. It is used to create a key without having to go through the entire install process.

By entering the serial number followed by the sequence, you open the part of the program where you enter the registration number without having to go through the complete install process.

FLKEYVAL

This is a protection utility usually run during installation; however, you can run it at any time. After you have the registration number, you must contact USDATA within ten days to obtain an authorization number. FLKEYVAL allows you quick access to the system to enter this number.

After turning on KEY, you are prompted for the authorization number without having to go through the entire install process.

- **FACTORYLINK UTILITIES**

- *FLSHM*

-
-

FLSHM

Use the `flshm` utility to

- List memory areas used by multi-user FactoryLink real-time databases.
- Clean up locked memory areas caused by abnormal shutdown of FactoryLink. This is only necessary in a multi-user environment.

The syntax for this command is

```
flshm [option] [rtdb_name]
```

where

option controls how `flshm` executes. This can be one or more of the following. If you do not specify any options, you receive a list of available real-time databases.

- L Lists existing real-time databases.
- U Lists all domain and user names for each real-time database.
- M Lists all shared memory segments for each domain.
- D Deletes the real-time database. Before using this option, stop the Run-Time Manager.
- C Clears the active flag for one or more user names for the indicated application and allows the user to start again without shutting down FactoryLink. Use this option only when an abnormal shutdown of a single domain occurs.

rtdb_name is the name of the real-time database on which to act. This field is required if you are using the `-D` option and optional with all the other options. If you do not specify *rtdb_name*, the actions are taken on the database defined in the `FLNAME` environment variable.

For example, specify the following command to unlock the shared memory areas for the `flapp1` real-time database.

```
flshm -lmu
```

UKEY

Use the UKEY utility to:

- List licensed options.
- Check licensed options.

Enter this command in lowercase if you are running in a UNIX environment. The syntax is

```
ukey [options]
```

where *options* can be one of the following. If you do not specify any options, a menu is displayed that includes all of the options available with *ukey*.

- l lists all the options provided with your license. If there are more options than can fit on one page, the list stops at the end of each page.
- lp lists all the options without stopping at the end of each page.
- d list all the options available. If there are more options than can fit on one page, the list stops at the end of each page.
- dp lists all the options without stopping at the end of each page.
- c checks if system is properly initialized. If the check is unsuccessful, the following message is displayed.

```
Key not installed
```

If the message above is displayed, install the software protection key or code.

- **FACTORYLINK UTILITIES**

- *Utility Messages*

-
-

UTILITY MESSAGES

If errors occur while FactoryLink is executing a utility, error messages are displayed on the screen. This section lists these messages, describes their cause, and provides suggested actions.

Can't open file *filename*

Cause: The system is unable to open an output file. The disk may be full, or the path may not exist.

Action: Delete any unnecessary files if the disk is full. Create the path if it does not exist.

Can't create file *filename*

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Can't create directory *directory name*

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Error writing to file *filename*

Cause: Either the operating system, third-party software, or hardware setup on your system is incorrect or incompatible, or an internal error has occurred.

Action: Verify the following:

1. The operating system is set up to run FactoryLink (tuning parameters, resources, etc.).
2. All third-party software needed by FactoryLink (such as X-Windows) has been installed and set up correctly and you have the correct version for FactoryLink.
3. All hardware is correctly set up and all of the hardware is compatible.

Contact Customer Support if everything is compatible and set up correctly, but the error continues to occur.

Out of RAM

Cause: No more memory is available.

Action: Close any unnecessary windows or programs. Add more memory to the system if this error occurs often.

- **FACTORYLINK UTILITIES**
- *Utility Messages*
-
-

FLNEW

The following error messages may occur during an FLNEW:

ERROR: FLINK environment variable is not set.

ERROR: Unable to find FactoryLink directory tree.

Cause: Either you have not set the FLINK environment variable or you have to set the FLINK environment variable to an invalid directory structure.

Action: Set the FLINK environment variable to a valid directory structure.

FLNEW aborted!

Cause: The FLNEW utility has stopped at your request.

Action: No action required.

The specified path, *path* does not exist.

Cause: The path you specified for the starter application does not exist or is invalid.

Action: Create a directory for the starter application and/or re-run FLNEW using the correct path.

EXPLODE

The following error messages may occur during an EXPLODE:

Error *number* in compressed file *filename*

Cause: An error was encountered during the file decompression process. The number in the message specified a particular system call to the compression library.

Action: This type of error is unrecoverable by the user. Contact Customer Support and provide the error message and number.

No RAM. Remove resident software and try again.

Cause: The decompression algorithm could not access enough buffer space due to memory used by the resident software.

Action: Remove any TSRs and try again. Contact Customer Support if this problem recurs.

Unable to open *filename* for compression/decompression.

Cause: The filename specified does not exist and cannot be opened for processing

Action: Verify the correct filename was specified. Re-enter the command.

Unable to open temporary file for compression/decompression.

Cause: The directory structure is full and a temporary file for the compression/decompression operations cannot be created.

Action: Remove unused files from the system. Re-enter the command.

FLSAVE

The following error messages may occur during an FLSAVE:

An error occurred while extracting application files. *FLTOOLS*

Cause: The multiplatform file may be corrupt or the hard disk may be out of space.

Action: The utility FLTOOLS provides an error message of its own. Follow the FLTOOLS onscreen instructions.

Backup failed!

Cause: FLXCOPY was unable to copy the application files to the destination path. Either you entered an invalid application file, or the files were corrupted.

Action: Check the usage to see if you have entered an invalid application filename. If so, re-enter the command using the correct filename. If you entered a valid application filename, manually save all application files to the \FLAPP directory by performing an X-COPY.

- **FACTORYLINK UTILITIES**
- *Utility Messages*
-
-

Backup to *filename* FAILED!

Cause: Either you entered an invalid destination path, or there may be a network error.

Action: Check the usage to see if you have entered an invalid destination path. If so, re-enter the command using the correct destination path. If you entered a valid destination path, there may be a network error. For network errors, use rcp (remote copying program) to copy the application to the desired node.

The directory *directory path* does not exist.

Cause: You specified a source path that does not exist.

Action: Re-enter the command with a valid source path.

Unknown third parameter

Cause: You have chosen an invalid third parameter.

Action: Re-enter the command using one of the following valid parameters:

- /RCP — to save to a remote node.
- /LOC — to save to a local drive.

Verification of remote copy capability failed.

Cause: FactoryLink is unable to remote copy to the destination file.

Action: Verify the network is configured to use rcp (remote copying program provided with TCP/IP software) to perform a remote copy to the specified host.

FLREST

The following error messages may occur during an FLREST:

A problem was found with file(s) by the file copy program *FLXCOPY*.

Cause: The files or application you are trying to restore may be corrupt or you may have entered an invalid filename.

Action: Retry the FLREST. Copy the application to the \FLAPP directory manually if that does not solve the problem.

An error occurred while extracting application files. *FLTOOLS*

Cause: The multiplatform file may be corrupt or out of hard disk space.

Action: The utility FLTOOLS provides an error message of its own. Follow the FLTOOLS onscreen instructions.

Cannot open Configuration file *source specification* /FLCONFIG.\$\$\$ for reading

Cause: You entered an incorrect source path.

Action: Retry the FLREST with a correct source path and a /LOC parameter to restore a multiplatform save.

Destination path is not full. Include drive with directory *i.e. C:\FLAPP*.

Cause: You did not enter a full path.

Action: Re-execute FLREST using a complete destination path (parameter 2).

Error # 7**Designated FLTAR file not in FLTAR format — FLTAR aborted.**

Cause: The FactoryLink file-extraction utility, FLTAR (FLTOOLS) (a transparent utility hidden within the FLREST utility) was given a file either corrupt or not a multiplatform-save file. The file might be a platform-specific save.

Action: Retry the FLREST without specifying a third parameter.

File FLINK *environment variable* missing from FactoryLink system!

Cause: A FactoryLink IV file is missing.

Action: Verify the FLINK environment variable contains the correct directory name. If so, obtain a copy of the file from the FactoryLink IV disk set.

Multi-platform application file *file* not found.

Cause: The system could not find the source file.

Action: Verify the filename of the first parameter is correct. Correct it if it is incorrect .

- **FACTORYLINK UTILITIES**
- *Utility Messages*
-
-

Restore failed!

Cause: Either a problem exists in the source or destination drive or the FLCONFIG.\$\$\$ list for FLXCOPY was lost.

Action: Check the usage to make sure the source and destination drives for the FLREST are correct. If either is incorrect, retry the FLREST using the correct source and destination drives. If both are correct, check that FLCONFIG.\$\$\$ exists in the FLAPP directory. If that file does not exist, copy it to the FLAPP directory. Then, manually X-COPY all application files to the FLAPP directory. FLREST need not be re-run.

Unable to copy application save file from remote node.

Cause: FactoryLink is unable to remote copy the source file.

Action: Verify rcp (remote copying program provided with TCP/IP software) can be run with the designated host and the source file exists.

Unknown third parameter

Cause: You have chosen an invalid third parameter.

Action: Re-enter the command using one of the following valid parameters:

- /RCP — to save to a remote node.
- /LOC — to save to a local drive.

FactoryLink Lite

FactoryLink Lite is designed to run small-scale FactoryLink applications that have been created in either a full FactoryLink or a FactoryLink Lite development package. This product supports a limited number of FactoryLink tags and offers all of the same features as the full scale run-time product.

Refer to FactoryLink Lite Product Matrix for current platform availability. The following requirements and limitations will help you to determine whether this product is suitable for your specific application.

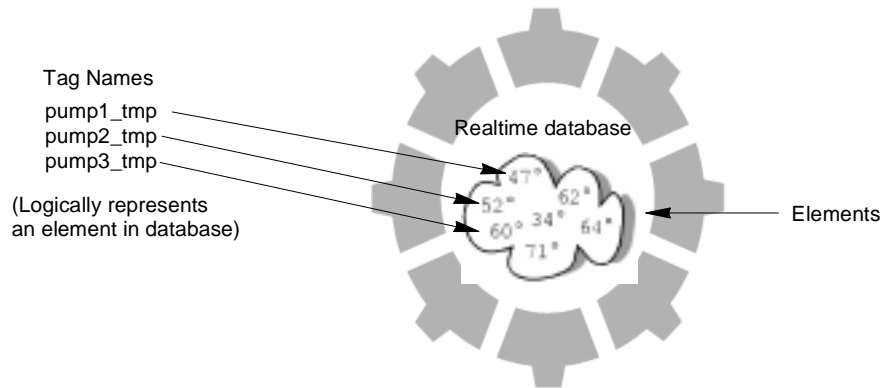
- **FACTORYLINK LITE**

- *Defining Tags*

-
-

DEFINING TAGS

Data stored in the real-time database is an element. Each element is assigned a logical name called a tag. This tag is used to reference the element in the real-time database.



Once an element is defined, you can make unlimited references to this element. Any FactoryLink task containing a reference to an element can read and write data to and from the element at run time.

During development, FactoryLink stores tag names in the FLAPP directory in the object database table. This information is updated to the .CT files when the run-time application is started.

Some tags are already defined in FactoryLink when it is shipped. Others are defined during application development either within the Configuration Manager or the Application Editor.

This chapter describes how to define tag names for database elements and provides some suggestions on how to use tags in your application.

TAG NAMING GUIDELINES

The following guidelines apply when you assign a tag name:

- 1-32 characters long
 - If using an array**, add delimiters of up to 16 characters.
Refer to “Defining Element Arrays” on page 207 for more details.
 - If using PowerNet**, use the model:
32 characters less 7 characters = 25 characters.
 - If using Scaling and Deadbanding**, use the model.
32 characters less 7 characters less 9 characters.
Refer to the *Application Editor Guide* for details on the added extensions.
- Valid characters are A-Z, 0-9, _, @, \$, ., :
- Do not start with a number
- No embedded spaces

System Added Extensions

When defining tag attributes in the Tag Definition dialog, new tags are created automatically from the originally defined tag name if you are using PowerNet or Scaling and Deadbanding.

PowerNet

When a new tag is created, an extension of up to 6 characters plus a dot (7 characters maximum) is added. These additional characters reduce the maximum length of the original tag name; therefore, the maximum effective length of the original tag name is 32 less 7, or 25. If the original tag name plus extension exceeds 32 characters, a warning is issued.

Scaling and Deadbanding

When a new tag is created, an extension of up to 8 characters plus a colon (9 characters maximum) is added. These additional characters reduce the maximum length of the original tag name; therefore, the maximum effective length of the original tag name is 32 less 7 less 9, or 16.

If you are not sure if you will be using PowerNet or Scaling and Deadbanding, you may choose to define tag names using only 16 characters. Also, remember if shortening tag names, do not reduce the length of the extensions.

- **FACTORYLINK LITE**
- *Developer-Defined Tag Element Maximum*
-
-

DEVELOPER-DEFINED TAG ELEMENT MAXIMUM

FactoryLink Lite is designed to process a limited number of developer-defined real-time database elements per application. All elements of tag arrays are counted toward this total. This total is in addition to the pre-defined elements provided with a new, blank application. The total number of tags available to you is determined by your licensing agreement.

Each time you configure an element during configuration, FactoryLink Lite adds it to the total count of elements you have defined for that application. If you define more elements than allowed, FactoryLink displays an error message.

At run time, FactoryLink Lite checks the application to determine the number of developer-defined elements. If the application has more than the licensed number of elements defined, the application will not run. If the application is started using “-d”, the error message can be seen in the Run-Time Manager window.

I/O Element Maximum

FactoryLink Lite contains I/O tasks. An I/O task is any task that communicates directly with an external device, such as a computer, a PLC, or nodes on a network.

These I/O tasks and their associated elements are a subset of the maximum number of elements you can configure for an application. FactoryLink Lite is designed to process up to the maximum number of I/O tag elements in all of the following I/O tasks:

- External Device Interface (EDI)
- PowerNet—Client only
- Local Area Networking (FLLAN)—both Send and Receive
- Dynamic Data Exchange (DDE)—Server only
- Various I/O protocol modules, such as AB NETDTL or Modicon Modbus Plus

An I/O element is any element that is the source or target of any of the following:

- A device I/O point in an EDI read/write table
- An external domain tag
- An FLLAN I/O address
- A DDE I/O address

For example, a FactoryLink element that is the target of a Modicon Modbus Plus register in an EDI read/write table is an I/O element. A FactoryLink element that triggers the read/write table is not an I/O element.

The effects of this maximum apply only to the loading of EDI, PowerNet, FLLAN, DDE, and other I/O tasks. At run-time, FactoryLink Lite checks the I/O tasks to determine whether any exceed the maximum number of licensed elements. If a task exceeds the maximum, the task shuts down.

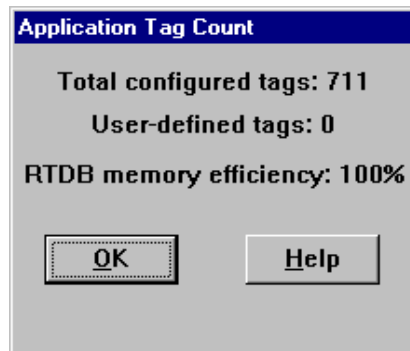
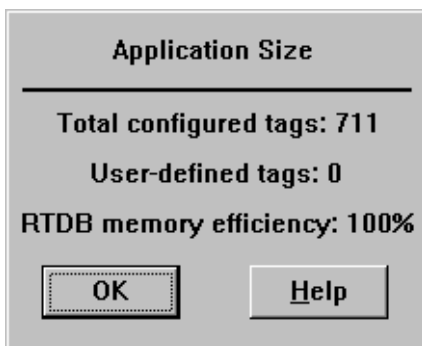
- **FACTORYLINK LITE**
- *Configuration Guidelines*
-
-

CONFIGURATION GUIDELINES

Configure a FactoryLink Lite application as you would any other FactoryLink application, but pay particular attention to the following items:

- Before you begin configuring, plan how you will use elements and I/O elements in the application so you do not reach the limit before you have defined all you need.
- Note the number of user elements available to you, as defined by your license agreement.

Choose Application> Size from the Configuration Manager menu bar or Application>Tag Count from the Application Editor menu bar, to display the Application Size dialog. The Application Size dialog is displayed.



This dialog displays both the total number of elements defined overall (including pre-defined elements) and the total number of elements you have defined. These totals cover all domains for the current application.

- Note the maximum number of I/O elements for I/O tasks.

The Application Size dialog does not separately display the number of I/O elements defined, although these are included in the total number of user-defined elements displayed in this dialog.

If you are developing a Lite application under a full-scale FactoryLink development package and you define too many elements, FactoryLink will not display an error message.

If you define too many elements or I/O elements, you must delete the extra elements or the application will not run. Complete the following steps to delete an element:

- 1 Choose View>X-ref List from the Configuration Manager menu bar to display the RTDB element cross reference panel. This panel lists all of the tasks that reference the element.
- 2 Delete the element from all configuration panels where the element is referenced.
- 3 Delete the element from all graphics objects.
- 4 Choose View>Object List from Configuration Manager to display the RTDB Element List panel.
- 5 Delete the element from the RTDB Element List panel.

- **FACTORYLINK LITE**
- *FactoryLink Lite Error Messages*
-
-

FACTORYLINK LITE ERROR MESSAGES

If an error related to Lite occurs while you are starting up or running a Lite application, one of the following messages may appear on the Run-Time Manager screen:

Application size exceeded by (number) tags for FLS-Lite

- Cause:** This error appears during development and can have either of the following causes:
The application has more than the allowable number of elements configured; this exceeds the Lite development system limit.
- Cause:** The application already has the maximum number of elements defined and you tried to define another one.
- Action:** Either do not edit or run the application on a Lite system, or, using a non-Lite development system, remove at least (number) elements from the application. Refer to “Configuration Guidelines” on page 424 for instructions for deleting elements. None-but, do not attempt to define more elements unless you delete some.

I/O Point count total exceeded

- Cause:** You have defined more than the maximum number of I/O total elements for the combined tasks.
- Action:** Delete some of the I/O elements. Refer to “Configuration Guidelines” on page 424 for instructions for deleting elements.

Lite Tag limit exceeded by (number) tags

- Cause:** This error appears at run time. You have defined more than the maximum number of elements.
- Action:** Delete some of the elements. Refer to Configuration Manager Main Menu for instructions for deleting elements.
- Cause:** If the Configuration Manager and Application Editor indicate you have defined more elements than the Run-Time Manager indicates at run time, then some of the elements you have defined are not used by any task. Delete these unused elements.

Cause: To delete unused elements:

1. Open the Configuration Manager.
2. Open the RTDB Element List panel.
3. Tab over to the Segment and Offset fields.
4. Scan the entries in these columns for blanks. These fields will be blank for unused elements.
5. Use Delete from the Edit menu to delete all elements whose Segment and Offset fields are blank.
6. Click on Enter to save the information.

When you re-start the application, the Run-Time Manager will display the same number of defined elements as the Configuration Manager and Application Editor.

- **FACTORYLINK LITE**
- *FactoryLink Lite Error Messages*
-
-

Chapter 111

Glossary

A

| | |
|--|--|
| AC file | See “attribute catalog” . |
| alarm group | Set of alarm conditions having similar characteristics that you configure in the Alarm Groups table. |
| Alarm Summary screen | On-line display of alarm information. |
| Alarm Supervisor | FactoryLink task that identifies alarm conditions and records events that occur during run-time FactoryLink operation. |
| alphanumeric | Alphabetical or numeric; an alphanumeric symbol is upper- or lower-case letters , or a digit (0 - 9). |
| analog | (1) Data type supported by FactoryLink; an analog element may assume any integer value from -32768 to 32767. (2) Signal of continuously variable voltage used as a measurement of some infinitely variable quantity, such as temperature, water salinity, blood pressure. |
| AND | Binary logical operator that yields a value of 1 (TRUE) if both of its arguments are non-zero and 0 (FALSE) otherwise. |
| animate | To define a graphics object so it changes color and/or shape or displays numeric values or text messages. |
| animation object | Graphically displayed input field, output field, or symbol created using the FactoryLink Application Editor. An animation object may be defined to change color or shape, accept various types of operator inputs, or display alphanumeric data. See Application Editor, display object, and input object. |
| animation testing | Feature in FactoryLink that enables testing each animation component in a drawing within the Application Editor and provides validation of complex animation types. |
| API (Application Programmer's Interface) | Set of utilities and/or procedures that allow an application-level program to access application-level functions. |
| application | Collection of information from configuration tables generated from the configuration databases and graphics for a set of FactoryLink tasks. |
| archiving | Creating a file of data records by logging information to a specified archival device, such as a disk drive. |
| argument | Numeric or string expression (operand) passed to an operator. |
| ASCII (American Standard Code | A seven-bit standard code used for information interchange among data processing systems, communication systems, and associated equipment. |

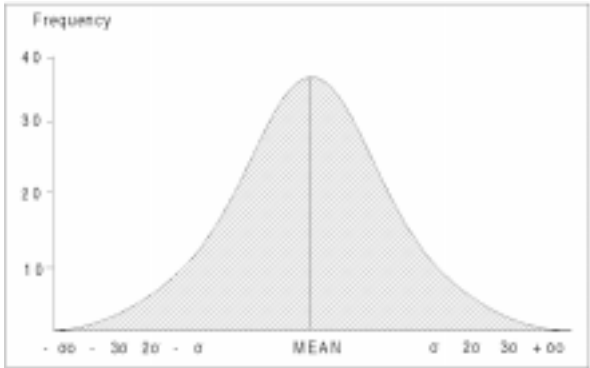
• **GLOSSARY**
•
•
•

| | |
|--------------------------|--|
| assignment statement | Mathematical expression that specifies a calculation to be performed with the results written to a FactoryLink real-time database element. |
| attribute | Assigned characteristic for a group of data points used in evaluating statistical information. |
| attribute catalog | ASCII text file that describes the database tables and the configuration information entered through the Configuration Manager and the Application Editor; also called an AC file. |
| attribute data (quality) | Data coming from yes/no, pass/fail determinations of whether the units conform to standards. May or may not include weighting by seriousness of defect. |
| average | See mean. |

B

| | |
|--------------------|---|
| background display | Graphics display created with the FactoryLink Application Editor that contains static (non-moving) graphics images. |
| background task | Non-interactive FactoryLink task that performs a specific set of functions that operate on FactoryLink data. |
| banner line | Single line display of information about a sorted alarm group. |
| Batch Recipe | Optional FactoryLink task that allows operators to transfer sets of values between the FactoryLink real-time database and disk files. |

bell-shaped curve A curve or distribution showing a central peak and tapering off smoothly and symmetrically to tails on either side. The following example is a normal curve..



Bell-Shaped Curve

| | |
|------------------------|---|
| binary number system | Numeric system used in computers with a base of 2; two digits (0 and 1) represent numerical quantities. |
| bit | Abbreviation for binary digit. The smallest unit of information, consisting of only two possible states, which may be 1 or 0, ON or OFF, TRUE or FALSE, or YES or NO, depending on the context. |
| block key | Hardware component plugged into a FactoryLink key ring that contains information used by FactoryLink to enable licensed software options. See button, key-ring, protection device. |
| block read (triggered) | Triggered read operation in which the EDI task directs the protocol module to read the values of all external device addresses specified in a single table and write them to the real-time database. |
| Block Read Complete | Digital element with a value force-written to 1 (ON) by the EDI task whenever any block read operation for the read table is completed. If this element is defined when the EDI task initializes, its value is force-written to 1 (ON). |
| Block Read Disable | Digital element of 1 (ON) that disables the block read of the elements specified in a read table. |
| Block Read Priority | Priority that influences the order in which the EDI task handles the queuing of block read requests. The highest priority (lowest number) is processed first. |

GLOSSARY

| | |
|----------------------------|---|
| Block Read State | Digital element with a value of 0 (OFF) when a block read of the elements specified in a read table is in progress and 1 (ON) when the table is inactive. When the EDI task initializes, the Block Read State element is force-written to 1 (ON). |
| Block Read Trigger | Digital element of 1 (ON) that initiates a block read of the values specified in a Read/Write Information panel. See trigger. |
| block write (triggered) | Write operation in which the EDI task directs the protocol module to transfer the values of all the elements specified in a Read/Write Table to an external device. |
| Block Write Complete | Digital element with a value force-written to 1 (ON) by the EDI task whenever any block write operation for the write table is completed. If this element is defined when the EDI task initializes, its value is force-written to 1 (ON). |
| Block Write Disable | Digital element of 1 (ON) that disables the block write of the elements specified in a write table. |
| Block Write Priority | Priority that influences the order the EDI task handles the queuing of block write requests. The highest priority (lowest number) is processed first. |
| Block Write State | Digital element with a value 0 (OFF) when a block write of the elements specified in a write table is in progress and 1 (ON) when the table is inactive. When the EDI task initializes, the Block Write State element is force-written to 1 (ON). |
| Block Write Trigger | Digital element with a value, when forced to 1 (ON), initiates a block write of the values specified in a Read/Write Information panel. See trigger. |
| boolean | (1) Digital. (2) A value of 0 or 1 represented internally in binary notation. |
| boot | Load an operating system onto a computer. |
| browse table | A configuration table specifying the correlation between elements and database information. |
| BUFSIZE | The size of a data buffer in networking. |
| button | Hardware component plugged into a FactoryLink button holder that contains information used by FactoryLink to enable licensed software options. See button holder, protection device. |
| button holder | Hardware device that contains information about the licensed FactoryLink options. See block key, button, key ring, protection device. |

byte Unit of storage consisting of eight bits. Because each bit has two possible states, a byte has 256 (2 to the 8th power) possible states.

GLOSSARY

C

c-chart For attribute data: A control chart of the number of defects found in a subgroup of fixed size. Use the c-chart when each unit typically has a number of defects.

Cp For process capability studies: C_p is a capability index that may range in value from 0 to infinity with a larger value indicating a more capable process. A value near 1.33 normally considered acceptable.

$$C_p = \frac{\text{Total Tolerance}}{6 \times \text{Sigma}} = \frac{USL - LSL}{6\sigma}$$

Cp Formula

where

USL = Upper specifications limit

LSL = Lower specification limit

σ = Standard deviation

Cpk For process capability studies: Capability index that measures how capable a process is of producing units within the specifications or tolerance limits. If the process is centered on the mean specification, Cpk has a value equal to C_p . If Cpk is negative, the process mean is outside the specification limits; if Cpk is between 0 and 1, some of the 6σ spread falls outside the tolerance limits. If Cpk is larger than 1, the 6σ spread is completely within the tolerance limits.

$$Cpk_l = \frac{\bar{\bar{x}} - LSL}{3\sigma}$$

-

$$Cpk_u = \frac{USL - \bar{\bar{x}}}{3\sigma}$$

-

$$Cpk = \min(Cpk_l, Cpk_u)$$

Cpk Formula

where

USL = Upper specifications limit

LSL = Lower specification limit

$\bar{\chi}$ = Grand average from control chart
(X-Bar)

| | |
|---------------------|---|
| CT | See configuration table. |
| CT file | See configuration table. |
| CTG file | See configuration table generator script. |
| CTGEN | <p>At run time the utility that generates the binary configuration tables (CT files) from the configuration databases.cell (of frequency distribution and/or histogram).</p> <p>An interval of the variable for which all the elements falling in that interval are summed together. Usually the full range of the variable is divided into cells of equal size and only the total number of elements falling into each cell is used in working with the frequency distribution and/or histogram. This greatly reduces the amount of information.</p> |
| centerline | For control charts: The horizontal line marking the logical center of the chart indicating the expected value of the quantity being charted. |
| change-read call | Read operation that, when called, checks the change-status flags of a list of elements in the real-time database. |
| change-status flag | Bit that indicates a change in the value of an element or whether the value has changed since the last time it was read. See exception processing. |
| change-wait call | Read operation that, when called, checks the change-status flags of a list of elements in the real-time database. |
| client | Task that sends a request for action to another task. |
| client-server model | A task operates as the client when it sends requests for action to another task (the server). The receiving task acts as the server when it responds to a request for action and sends a reply. |

• GLOSSARY

| | |
|--------------------------------------|---|
| cold start | When starting a domain instance, the initialization of all elements including persistent elements to their default values found in the Configuration Manager. See persistence, persistent elements, and warm start. |
| column | (1) Part of a relational database record (a record is composed of one or more columns). Also called a field. (2) A vertical group of data entry fields of the same type. |
| common causes | Those sources of variability in a process which are truly random or inherent in the process itself. |
| completion trigger | Element that is force-written to a 1 (ON) when an operation is complete. See trigger. |
| composite object | Two or more graphic objects combined so the system treats them as a single object. See simple object, subobjects. |
| conditional | Depending on or relating to a condition(s). |
| conditional statement | See control statement. |
| configuration database | Relational database that stores FactoryLink configuration data. |
| Configuration Manager | FactoryLink development tool that allows you to define the functionality of each task in an application by specifying information in a configuration table(s). |
| configuration mode | Mode of operation that the FactoryLink system is in when you set up an application. In the configuration mode, you use the Application Editor or the Configuration Manager. |
| configuration table | Binary file, or CT file, that the CTGEN utility produces at run time that contains data extracted from the configuration database table. |
| configuration table generator script | Script file, or CTG file, that tells the CTGEN utility how to extract data from a database table and combine the extracted values to produce a binary configuration table (CT) file at run time; found in /{\$FLINK}/CTGEN . |
| constant | An unchanging quantity or any numeric or string expression that contains no variables. |
| continuous data | Data for a continuous variable. |
| continuous variable | A variable which can assume any of a range of values; an example would be the measured size of a part. |

| | |
|--------------------------------------|--|
| control (of process) (statistical) | A process that exhibits only random variations. |
| control (of process) (manufacturing) | A process with variations within specified control limits. (Not related to statistical control). |
| control chart | Graphic representation of a parameter of process performance used to determine if parameter is within acceptable quality limits. |
| control limits | The limits within which the product of a process is expected (or required) to remain. If the process exceeds the limits, it is said to be out of control. (Not the same as tolerance limits). |
| control panel | See panel. |
| control statement | Mathematical expression that includes instructions about the circumstances under which a block of code is to be executed. Also called conditional statement. |
| cursor | Light indicator on a monitor that shows where the next character is to be generated. The cursor can be moved across the screen by the use of a key on the keyboard or with a pointing device. Also refers to the chart cursor which points to a specified instant in time on a Trend or SPC plot. The chart cursor retrieves the numeric value or values which are plotted for that instant in time. |
| cycle | Recurring pattern. |

D

| | |
|------------------|--|
| Database Browser | FactoryLink task that uses SQL statements to communicate with a Historian; sends and retrieves data to and from external database tables, including tables created outside of FactoryLink. |
| database element | See element. |
| database file | Group of related database records. |
| Database Logger | FactoryLink task that enables other FactoryLink tasks to send element information to a database historian. |
| database table | FactoryLink configuration database table that stores information generated by a task from the Main Menu or data in the task's panels. |
| data type | Digital, analog, long analog, floating-point, message, or mailbox data types supported by FactoryLink. |

• GLOSSARY
•
•
•

| | |
|-----------------------|---|
| decimal number system | Conventional base-10 numbering system. |
| default | Choice made by a program in the absence of a specific selection. |
| developer | Person who designs and configures a FactoryLink application. |
| Device Name | Name of an external device FactoryLink uses to communicate through a particular logical port. |
| digital | Numeric data type supported by FactoryLink; each digital element holds one bit of information of two values: 1 (ON/TRUE) or 0 (OFF/FALSE). |
| directory | List of files found in a particular area of the operating system. |
| discrete | Consisting of unconnected, distinct parts. |
| display object | Graphically displayed output field using the Application Editor that provides an area on a screen where the FactoryLink real-time database displays information output. See animation object. |
| domain | Characteristics of elements in the real-time database in either the USER or SHARED domain. See shared domain, user domain. |
| dynamic | Characterized by continuous change, activity, or progress. |

E

| | |
|---------------------------------|---|
| EDI architecture | Layered communication interface that supports communication links with COMM ports, terminal servers, and coprocessors. |
| EDI base module | An EDI architectural layer that insulates the application from the device and monitors and controls external processes. |
| EDI (External Device Interface) | FactoryLink task that allows information from an external device, such as a programmable logic controller (PLC), RTU, loop controller, distributed I/O, data-collection terminal, or bar-code scanner to be transferred to and from the FactoryLink real-time database. |
| editor | Computer program that creates, changes, manipulates, and deletes objects of a given kind. |
| electronic key | See button, block key. |
| electronic protection device | See protection device. |
| element | Data structure occupying memory in the FactoryLink real-time database. |

| | |
|---------------|--|
| element name | Name you assign to a real-time database element used during development of an application. |
| error message | Display of ASCII text or numeric code indicating that the system has detected an abnormal situation or incorrect data. |
| event timer | Digital element that initiates or controls a system function at run time whose value is forced to 1 (ON) no more than once every twenty-four hours (for example, Monday at 8:00 am). |
| EWMA Chart | Graphic display of a statistic that gives less and less weight to data as the data gets older and older. A plotted point on an EWMA chart can be given a long memory, thus providing a chart similar to the ordinary CUSUM chart; or it can be given a short memory, thus providing a chart analogous to a Shewhart (X-Bar) chart. |

$$EWMA = \hat{y}_t + \lambda(y_t - \hat{y}_t)$$

where

- y_t = Observed value at time t
- \hat{y}_t = Predicted value at time t (oldEWMA)
- $e_t = y_t - \hat{y}_t$ = Observed error at run-time of t
- $\lambda = (0 < \lambda < 1)$ = Depth of memory of EWMA

| | |
|----------------------|---|
| exception processing | Basic FactoryLink architecture whereby a real-time database element is read/written only when its value has changed. See change-status flag. |
| exception write | Write operation that transfers only the values of elements that have changed since the last time a task scanned the real-time database. See exception processing. |
| exponential constant | Mathematical expression consisting of an optional minus sign (-) followed by consecutive digits (0 - 9), an exponential operator (E), an optional minus sign (-), and another group of consecutive digits. Exponential constants are interchangeable with floating-point constants. |
| expression | Mathematical statement that is resolved to a value. |

F

• GLOSSARY

•

•

•

•

| | |
|---------------------------|--|
| FactoryLink application | See application. |
| FactoryLink kernel | See kernel. |
| FactoryLink SPC Run Rules | Tests that the SPC Data Monitor process uses to detect patterns of quality-level variations over time. The following shows the chart zones on which the FactoryLink SPC Run Rules are based. |



| Rule | Definition |
|-------------|--|
| 2 OF 3 | 2 out of 3 points lie on one side of the Center Line in Zone A or beyond |
| 4 OF 5 | 4 out of 5 points lie on one side of the Center Line in Zone B or beyond |
| TREND | 7 points in a row are rising or falling |
| 8CONJ | 8 points in a row lie on one side of the Center Line |
| 3SIGMA | one or more points lie outside of control limits |
| 15CONS_IN_C | 15 consecutive points lie within either upper or lower Zone C on the same side of the Center Line |
| 90%25_IN_C | of 25 points, at least 90 percent of these (or 23) lie within Zone C (Upper or Lower) and all 23 are on the same side of the Center Line |
| 8CONS_OUT_C | 8 consecutive points lie outside either Zone C and are all on the same side of the Center Line |
| 60%25_OUT_C | of 25 consecutive points, at least 60 percent (or 15) lie outside Zone C and are all on the same side of the Center Line |

| | |
|-------------------------|--|
| FactoryLink task | See task. |
| FALSE | Logical level associated with the numeric value of zero; logical operators return a zero when the result of their operation is FALSE. |
| field | Space in a configuration table or on a display that may or may not contain data; part of a record (a record is composed of one or more fields). |
| file | Specific set of data defined by the operating system. |
| File Manager | FactoryLink task that performs basic file-management operations, such as copying, printing, renaming, typing, and deleting files locally or remote. |
| file name | Name of a specific set of data . |
| file-transfer operation | Transmitting the contents of a file from one FactoryLink station on a network to another FactoryLink station on the same network using the File Manager task. |
| FIRST | Option for displaying an on-line alarm; displays the first (oldest) alarm on the system. |
| flag | Attribute that has only two possible states: ON or OFF. |
| FLAPP, {FLAPP} | FactoryLink environment variable that corresponds to the name of the directory structure containing the application-related files. If you use {FLAPP} in the pathname with its { }, the system uses the default environment variable. |
| FLCONV | FactoryLink utility that converts older applications to run under the current version. |
| FLDOMAIN, {FLDOMAIN} | FactoryLink environment variable specified during installation as the name of the domain under which the application is designed to run. There is no default for this variable. |
| FLINK, {FLINK} | FactoryLink environment variable that corresponds to the name of the directory structure containing the FactoryLink software system program files. If you use {FLINK} in the pathname with its { }, the system uses the default environment variable. |
| FLNAME, {FLNAME} | FactoryLink environment variable specified during installation as the name of the application. There is no default for this variable. |
| FLNEW | FactoryLink application utility that creates the user-specified directory structure and copies files required for development of a new application to this directory structure. |

GLOSSARY

| | |
|----------------------------|--|
| FLOAT | See floating-point. |
| floating-point | Data type supported by FactoryLink; each floating-point value occupies 10 bytes of storage in the real-time database and holds numeric values in the range +/- 1E-308 to +/- 1E308 with a precision of 15 decimal digits (all figures approximate). Abbreviated FLOAT. |
| floating-point constant | Mathematical expression consisting of an optional minus sign (-) followed by consecutive digits (0 - 9), a decimal point, and another group of consecutive digits. Floating-point constants may be used interchangeably with exponential constants. |
| FLOPT, {FLOPT} | FactoryLink environment variable that points to the location of the license information directory {FLINK}\OPT . The {FLOPT} directory contains the files FL.DEV (key type) and FL.KEY (options and license information). If you use {FLOPT} in the pathname with its { } , the system uses the default environment variable. |
| FLREST | FactoryLink application utility that restores and creates application files into a platform-dependent application. |
| FLSAVE | FactoryLink application utility that backs up graphics files, symbols, configuration tables, recipes, logs, and other files and creates a backup of the specified application. |
| FLUSER, {FLUSER} | FactoryLink environment variable specified during installation as the name of the domain instance. There is no default for this variable. |
| font | Complete set of type of one size and style. |
| forced-write | Write operation that writes a new value (even if it is the same as the old value) to an element and sets all of that element's change-status flags to 1 (ON). |
| formula | See statement. |
| frequency distribution | Number of times each outcome of a statistical population is observed. |
| FUNACK | Option for displaying an alarm on-line; displays the first (oldest) unacknowledged alarm. |
| function | Subroutine that performs a specified operation or calculates a specified result. Frequently activated by a keyboard input; hence, function key. See mathematical function. |

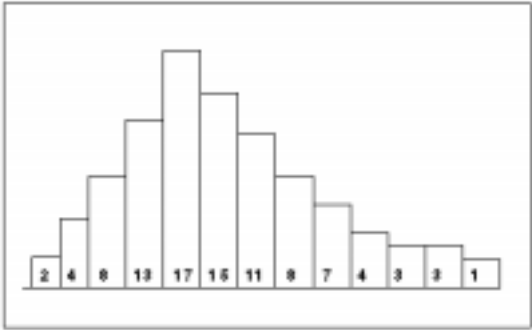
- Function Key** An animation object in the FactoryLink Application Editor that animates a key on the keyboard so it causes an action to occur when pressed at run time.
- function key** (1) A key used to send a signal to the computer program controlling the display; (2) a key that performs a specific set of operations.

G

- GFIRST** Option for displaying an on-line alarm; displays the first (oldest) alarm in a selected alarm group.
- GHIGH** Option for displaying an on-line alarm; displays the oldest and highest priority alarm in a selected alarm group.
- GLAST** Option for displaying an on-line alarm; displays the last (most recent) alarm in a selected alarm group.
- global element** See pre-defined element, reserved element.

H

- hexadecimal number system** Numeric system used in computers with a base of 16 in which the letters A-F represent numerical quantities equal to 10 through 15 in the decimal numbering system.
- Histogram** A graphic representation of a frequency distribution plotted by ranges of variables divided into equal intervals along with the number of observations accumulated in each.



Histogram

- Historian** FactoryLink task that provides a common interface with one or more database managers.

GLOSSARY

I

| | |
|----------------------------|---|
| icon | Graphical symbol representing a function that can be performed. |
| Individual | Type of real-time and historical control chart found in FactoryLink SPC. See X-Bar, EWMA. |
| information message | Onscreen display of ASCII text or numeric codes that describe normal system status, usually during system startup or shutdown that requires no user or operator action. Also called status message. |
| information panel | See panel, initialize. Set up a system for use. |
| input object | Graphically displayed input field created using the FactoryLink Application Editor. See animation object, input text field, pushbuttons. |
| input text fields | Area on a graphics display defined in the Application Editor to accept operator input in the form of text or numeric values. See input object. |
| instability (of a process) | A process that exhibits variations larger than its control limits or a systematic pattern of variation. |
| instance | A copy of a FactoryLink application that can be executed at run time. The number of instances allowed is the number of users that can simultaneously interact with the run-time system. See domain. |
| integer constant | Mathematical expression consisting of an optional minus sign (-) followed by consecutive digits (0 - 9). |
| interval timer | Element whose value is forced to 1 (ON) at least once every twenty-four hours at regular intervals of the system clock (for example, every 60 seconds). An interval timer can be used to initiate or control a system function at run time. |

K

K For process capability studies: a measure of difference between the process mean and the specification mean.

$$K = \frac{(Mean - Midpoint)}{(Tolerance / 2)}$$

K Index Formula

- kernel FactoryLink software module that creates the real-time database when the FactoryLink Run-Time system is started that provides security for the FactoryLink System and exchanges data among tasks.
- key See block key.
- keyboard cursor See text cursor.
- key ring Part of an electronic protection system that plugs into a FactoryLink key ring. Contains information used by FactoryLink to enable the licensed software options. See option key, protection device, button.
- Keystroke Keyboard key or button that is being animated using the Application Editor.
- keyword files ASCII text files that tell the Configuration Manager how to translate text table entries into binary values; also referred to as key files found in the /{**FLINK**}/KEY directory.

L

- LCL Lower Control Limit. For control charts: the limit below which the quality of a process is out of control.
- LSL Lower Specification Limit. The lowest value of a product dimension or measurement which is acceptable.
- ladder logic Language used to configure programmable logic controllers; industry standard for representing relay-logic control systems.
- LAN See Local Area Network.
- library Collection of utility functions that primarily interface application and system programs to the FactoryLink kernel.
- inking Process of associating an object and its animation attributes to elements in the FactoryLink real-time database.

•
•
•
•
GLOSSARY

| | |
|--------------------------|--|
| Local Area Network (LAN) | In-house data-communications system connecting a number of microcomputers. See node, node name. |
| LOCAL file | ASCII file in which you define a local station name. The LOCAL file resides in the /{ FLAPP }/NET directory. |
| local station | Current station on a network. |
| log | Set of data files, error messages, or alarms spooled to a printer or archived to disk. |
| logical operator | Symbols in expressions that test operands for TRUE (non-zero) or FALSE (zero) values and return a result of 1 (TRUE) or 0 (FALSE). |
| logical port | Number you enter in the FactoryLink External Device Definitions Table to represent the physical port. When using an IBM RIC/ARTIC card, this number represents a combination of the physical card and the physical port. |
| logical station | Number that you enter in a Read or Write Table representing the combination of a logical port with a physical station. |
| LONGANA | Abbreviation for long analog. |
| long analog | Data type of a 32-bit signed integer supported by FactoryLink. Abbreviated LONGANA. |

M

| | |
|-----------------------|--|
| mailbox | Data type supported by FactoryLink; a mailbox element is organized as a queue of mailbox messages and associated message data and consists of variable length and variable structure. |
| mask | Prevents an alarm from being activated. |
| Math & Logic | FactoryLink task that performs mathematical and logical calculations and assigns the results to elements. |
| mathematical function | (1) A mathematical expression describing a relationship between two or more variables. (2) A function that implements a mathematical operation, such as square root or cosine. For a list of mathematical functions, refer to the <i>FactoryLink Configuration Guide</i> . |

Mean (of a statistical sample) The average value of some variable. The mean is given by the following formula where x is the value of the variable to the i th element and n is the number of elements in the sample.

$$\bar{X} = \frac{x_1 + x_2 + \dots + x_n}{n}$$

Mean

- menu Set of choices for functions displayed together overlaying the present contents of the screen without disrupting them.
- message Data type of string or binary data having a total length of 64K supported by FactoryLink.
- monochrome Type of monitor that displays information in only one color.
- mouse cursor Small arrow or shape in the Application Editor representing a selected option that indicates the point on the screen where the next keyboard input or mouse click is displayed. A pointing device moves the mouse cursor.
- moving average/moving range Same as X-Bar and range calculation except that a circular buffer of Moving Size is used instead of Subgroup Size.
- moving average/moving range chart A chart used in FactoryLink SPC having points which are averages or ranges of previous data carried forward with the current data. Used to dampen the effects of a single reading when only one reading is made per time period or when a process is strongly linked to a previous data output. See range, X-Bar.

N

- NETBIOS Software standard used for local area networks or optional FactoryLink task used for communicating between FactoryLink systems over a local area network.
- network Group of terminals and one or more computers connected for moving information from place to place.
- node Station or general junction point on a Local Area Network (LAN). See Local Area Network.
- node name Unique name for a node on a specific Local Area Network (LAN). See Local Area Network.

• GLOSSARY
•
•
•

| | |
|---------------------|--|
| NOT | Unary logical operator that yields a value of 1 (TRUE) if its argument is 0 (FALSE) and 0 (FALSE) if its argument is TRUE (nonzero). |
| np-Chart | For attribute data: a control chart of the number of defective units in the subgroups with fixed subgroup sizes. |
| NUMBUFS | Number of data buffers in which data can be queued In networking. |
| numeric constant | Integer or floating-point (or exponential) constant. |
| numeric input field | Area on a graphics display defined in the Application Editor to accept operator input in the form of a number. See input text field, input object. |

O

| | |
|---------------------------------|---|
| object | See animation object. |
| OBJECT database table | Configuration Manager-maintained database table that stores the real-time database definition and other related information. |
| octal number system | Numeric system with a base of 8 used in computers. |
| Open Software Bus architecture | FactoryLink operating principle whereby modular software programs run concurrently and communicate through a global real-time database. |
| operand | See argument. |
| operating characteristics curve | For acceptance sampling: curve showing the percent defective in a lot vs. the probability that the lot will be rejected for a specified lot size and sampling plan. |
| Operating System/2 (OS/2) | Prioritized, multitasking environment that manages system resources, such as memory, disk drives, input/output devices, printer ports, and other system components. Abbreviated OS/2 [®] . |
| operator | (1) The person who operates the finished application. (2) Mathematical function. The operators supported by Interpreted Math & Logic include the most common arithmetic, logical, relational, and conditional operators used in computer science and related disciplines. |
| option key | See block key ,button. |
| OR | Binary logical operator that yields a value of 1 (TRUE) if at least one of its two arguments is TRUE (non-zero), and 0 (FALSE), otherwise. See operator. |

OS/2[®] See Operating System/2.

out of control Variations larger than the quality control limits.

P

PAK See Programmer's Access Kit.

p-chart (percent defective) For control charts: the percentage of defective units in which the subgroup size varies. Used for attribute quality control.

PLC See Programmable Logic Controller.

panel Screen display that provides pre-defined entry fields for information required by a configuration table.

Pareto analysis An analysis of the frequency of occurrence of various possible concerns. This is a useful way to decide quality control priorities when more than one concern is present. The Pareto Principle states that 20% of concerns cause 80% of the problems.

Pareto diagram A bar graph showing the frequency of occurrence of various concerns with the most frequently occurring ones first and with a line chart showing cumulative data vs. data type.

path name Route of a sequence of file names or directory names for an operating system to follow to locate information stored in a computer.

pen Element whose value represents a particular combination of attributes for trending activity. Contains values used in an on-line trending chart the same way values recorded by a pen are used in a strip-chart recorder.

pen trace Graphic representation of a record of trending activity associated with a particular pen. On an on-line trend chart, resembles the line drawn by a pen on a strip-chart recorder.

percent defective For acceptance sampling: the percentage of defective units or units of unacceptable quality in a lot.

percent specifier See variable specifier, sprintf string.

persistence Ability of an element to maintain its value over an indefinite period of time.

persistent elements Real-time database elements that maintain their value when a domain instance is closed, either deliberately or inadvertently (from a power loss or faulty process). This is possible because the values of the persistent elements are written to permanent storage and are not affected by a system shut down.

• GLOSSARY

•

•

•

•

| | |
|---------------------|--|
| physical station | Physical address of an external device on a network. |
| pixel | A group of light phosphors that can be stored, addressed, or displayed by a computer. Abbreviation for picture element. |
| pointing device | Device, such as a keyboard or a mouse, that allows control over movement of the cursor during the creation of graphics images and displays. |
| polygon | Closed-plane figure bounded by three or more line segments. |
| Power Edit | An editing tool that enables you to select and directly edit subobjects, including their animation, within a composite object without breaking apart the composite. See composite object, simple object, subobject. |
| Power Objects | User-extensible, animated compound objects that contain template variables for the animation features. See Power Packs. |
| Power Packs | Libraries of drawings where you can store Power Objects for reuse in creating new instances of similar objects in the same or in other applications. See Power Objects. |
| pre-defined element | Convenient element configured for internal system use. Sometimes called reserved element or global element. |
| Print Spooler | FactoryLink task that extends the printing capabilities of the system by providing multiple print buffers and the ability to use serial or parallel printers. Also allows output to be directed to a disk file. |
| priority | <p>(1) Three-digit hexadecimal number, such as 201, that specifies the processing priority for a FactoryLink task.</p> <p>where</p> <p>the first number (2 in 201) specifies the operating system class of priority.</p> <p>the second number (01 in 201) specifies the priority within the class listed above.</p> <p>The higher the number, the higher the priority within the class.</p> <p>(2) Rank or position of an operation in a sequence of operations.</p> |
| process capability | The level of uniformity of product that a process is capable of yielding expressed by the percent of defective products or the range or standard deviation of some product dimension. Process capability is usually determined by performing measurements on product units produced by the process. |

| | |
|-------------------------------------|--|
| process control | Maintaining the performance of a process at its capability level by sampling the process product, charting its performance, determining causes of any problems, and taking corrective actions. |
| Programmable Counters | FactoryLink task that provides triggered up/down counters with terminating triggers. |
| Programmable Logic Controller (PLC) | Special purpose computer programmed to control process or machine operation. The programmable logic controller consists of five basic components: processor, memory, input/output, power supply, programming device. A PLC is designed as an industrial control system. |
| Programmer's Access Kit (PAK) | Collection of optional FactoryLink software tools and programming libraries and keys related documentation for use by programmers in their design and construction of FactoryLink-compatible programs. |
| prompt | Character or message on a video display indicating the system is ready to accept a command. |
| protection device | A button in Windows NT, Windows 95, or OS/2 FactoryLink systems plugged into a button holder or one or more electronic block keys that are plugged into a key ring. The button holder and key ring plug into the parallel port of the printer. The protection devices contain information FactoryLink uses to enable icensed software options. |
| protocol module | An EDI architectural layer that insulates the EDI base module from the type of external device you are using and provides the flexibility to support many different types of external devices. |
| pushbuttons | Input objects used with a pointing device or function key that performs an operation, such as shutting down a task. |

Q

| | |
|------------------------|--|
| quality characteristic | Particular aspect of a product that relates to its ability to perform its intended function. |
|------------------------|--|

R

| | |
|---------|---|
| R-Chart | A control chart of the range of variation among the individual elements of a sample or the difference between the largest and smallest elements as a function of time, lot number, or similar chronological variable. |
|---------|---|

• GLOSSARY

•

•

•

•

range (1) The difference between the highest and lowest of a group of values. (2) Type of real-time and historical control chart found in FactoryLink SPC. See X-Bar, moving range.

$$R = \frac{\text{(Largest value in Subgroup)}}{\text{(Smallest value in Subgroup)}} \text{ minus}$$

read call Operation that returns the value of an element regardless of the value of the element's change-status flag.

read complete See Block Read Complete.

read disable See Block Read Disable.

read priority See Block Read Priority.

read state See Block Read State.

read trigger See Block Read Trigger.

real-time Immediate software response to an event.

real-time database Memory-resident array of information that acts both as an in-memory storage device and an interprocess communication mechanism for FactoryLink tasks. All FactoryLink tasks share the information in the real-time database by reading from or writing to real-time database elements.

Real-Time Database Debugger FactoryLink task that allows you to modify the real-time database during run time and to monitor process access to the real-time database; used to develop and debug applications. (RTMON).

real-time database element See element.

reboot Boot again. See boot.

Recipe See Batch Recipe.

record In a relational database, a unit or set of data that is the basic component of a file. Also called a row.

relational operator Mathematical function that yields a TRUE or FALSE result that depends on the signs and relative magnitudes of its arguments. The FactoryLink Interpreted Math & Logic program recognizes six relational operators:

< <= > >= = !=

| | |
|------------------------|---|
| relational subgrouping | For control chart: subgroup of units selected to minimize the differences because of assignable causes. Samples taken consecutively from a process operating under the same conditions usually meet this requirement. |
| relay | Electrically operated device that mechanically switches electrical circuits. |
| relay logic | Representation of a program or other logic in a form normally used for relays. |
| remote station | Other station on a network. |
| Report Generator | FactoryLink task that formats real-time database information into a predefined, field- printed report. |
| reserved element | See predefined element. |
| result table | Temporary table containing data selected from an external database. |
| row | See record. |
| run | A set of consecutive units sequential in time. |
| run time | Period during which a FactoryLink application is operating. |
| Run-Time Graphics | The FactoryLink Graphics task. Animated displays with which the operator interacts when the application is running. For this task to be available, the Application Editor must be installed. |
| run-time mode | State of the FactoryLink system when an application is operating. |
| Run-Time Manager | FactoryLink task that supervises all other FactoryLink tasks as they operate. |

S

| | |
|---------------------|--|
| sample (statistics) | A representative group selected from a population used to determine the properties of the population. |
| sample size | The number of elements, or units, in a sample. |
| sampling | The process of selecting a representative sample of a population and determining the properties of the sample. |
| sampling variation | The variation of a sample's properties from the properties of the population from which it was drawn. |
| select object | See input select object. |

• GLOSSARY

•

•

•

•

| | |
|--------------------------|--|
| server | (1) Task that receives requests for action from other tasks. (2) A computer that provides a shared hard disk and possibly other resources in a local area network. |
| SHARED domain | Collection of elements in the real-time database that contain publicly-owned data and tasks available to all users at run time. See domain, USER domain. |
| Sigma | (1) Standard deviation of a statistical population. Represented by the symbol σ . (2) Type of real-time and historical control chart found in FactoryLink SPC. See X-Bar, moving averages. |
| sigma limits | For histograms: lines marked on the histogram showing the points n standard deviations above and below the mean. For control charts: control lines evenly spaced on either side of the center line. |
| signals | Notifications of events used to affect process control. |
| simple object | Single objects, such as lines, circles, and text, not composed of any other objects. The basic building blocks of a graphics display. See composite object, subobjects. |
| sprintf string | Group of ASCII characters that specify a standard format for one or more variables to be inserted in a character string preceded by the % symbol. |
| stability (of a process) | A process showing no recognizable pattern of change. See control, constant cause system. |
| standard deviation | A measure of the variation among the members of a statistical sample. If a sample of n values has a mean of \bar{x} , its standard deviation is given by the formula shown. |

$$\sigma = \sqrt{\frac{\sum (x - \bar{x})^2}{n - 1}}$$

Standard Deviation Formula

| | |
|---------------|---|
| start trigger | Element whose value, when forced to 1 (ON), initiates an action in a FactoryLink task. |
| statement | Mathematical expression or formula that may contain any of the following components. <ul style="list-style-type: none">• Directives• Expressions, including operators, variables, and/or constants |

| | |
|---------------------------------------|--|
| | <ul style="list-style-type: none">• Comments• Functions |
| station | Any PLC, computer, or data terminal connected to and communicating through a network. |
| station name | Name given to a device connected to a network. |
| statistical control (of a process) | A process that exhibits only random variations. |
| status message | See information message. |
| string constant | ASCII numeric expression consisting of an opening double quotation mark followed by zero or more printable characters and a closing double quotation mark. |
| subdirectory | Directory contained within another directory. |
| subgroup | For control charts: sample of units taken at or near the same time from a given process. |
| subobjects | Each object in a composite object is called a subobject. Subobjects can contain simple and/or composite objects. See composite object, simple object. |
| symbol | A graphic object exchanged with another object during runtime. |

T

| | |
|------------|--|
| table | See configuration table, database table. |
| tag name | See element name. |
| tag number | A two-part number FactoryLink assigns to an element and uses to refer to that element when writing a value to it or reading its current value. For example, the tag number 0:177 |
| where | <div>0 is the segment (block of space in the real-time database that stores elements of one datatype 177 is the offset (location of the bits in the segment that contain the element).</div> |
| task | FactoryLink program that reads from and writes to the real-time database and performs a specific function. |

• GLOSSARY

•

•

•

•

| | |
|------------------------------|---|
| task-specific database table | Database table that stores information related solely to a specific task. |
| text cursor | Flashing vertical bar that indicates the point on the screen where the next keyboard input will be displayed. |
| Timer | FactoryLink task that updates real-time database information based on the chronological events and time intervals specified in its configuration tables. |
| toggle | To switch between two states or values. |
| tolerance | The permissible range of variation in a particular dimension of a variable. Tolerances are often set by engineering requirements to ensure components will function together properly. |
| trend | A gradual, systematic change with time or another variable. |
| Trending | FactoryLink task that displays data stored in a relational database as a strip chart on a real-time graphics display. |
| trending chart | FactoryLink graphics display that resembles a strip-chart recorder. Composed of a chart object and pen objects that function in combination with real-time database values for on-line display of real-time and/or historical data. |
| trigger | Digital element that causes an event to occur when its value is forced to 1 (ON). |
| TRUE | Logical level associated with any non-zero numeric value, usually 1 (ON). |
| type | See data type. |
| TYPE database table | Database table that defines segments where a data type used is stored in the real-time database. |

U

| | |
|---------|--|
| u-Chart | For attribute data: a control chart of the percentage of defects in one inspection subgroup of non-fixed size. |
| UCL | Upper Control Limit. For control charts: the limit above which the quality of a process is out of control. |
| UKEY | FactoryLink utility you view information in the software security mechanism with. |
| USL | Upper Specifications Limit: the highest value of a product dimension or measurement which is acceptable. |

| | |
|------------------|--|
| unary operator | Mathematical function that takes one argument, usually placed after it. For instance, in the expression NOT Overflow, the logical negation operator NOT has Overflow as its argument. |
| unsolicited read | Command issued by an external device, independently of FactoryLink, to write information to the real-time database without requesting the data. |
| USER domain | Collection of elements in the real-time database that contain data and tasks available locally to a specific user at run time. In a multiuser system, each user owns a copy of all the elements in the USER domain. See domain, SHARED domain. |

V

| | |
|--------------------|--|
| variables | Quantities subject to change or variability. |
| variable data | Concerning the values of a variable as opposed to attribute data. |
| variable specifier | ASCII character that represents a replaceable value in a character string. See sprintf string. |
| variance | The square of the standard deviation. |
| VERBOSE | A command-time option that flags the process to provide debugging information. |

W

| | |
|---------------------|---|
| warm start | When starting a domain instance, the initialization of all non-persistent elements to their default value. Similar to a cold start and restores all persistent elements to their previously saved values. See cold start, persistence, persistent elements. |
| where clause | SQL (Structured Query Language) statement part that specifies a filter for accessing data in a relational database. |
| wild-card character | Symbolic character, usually an * (asterisk), taken to mean any character in a character string. |
| window | Area occupying either part or all of a screen on which program information is presented. |
| write call | Operation that writes a new value to an element and, if the new value is different from the previous value, sets all of that element's change-status flags to 1 (ON). |
| write complete | See Block Write Complete. |
| write disable | See Block Write Disable. |

• GLOSSARY
•
•
•

- write priority See Block Write Priority.
- write state See Block Write State.
- write trigger See Block Write Trigger.

X

X-Bar (1) Mean or average represented by the symbol \bar{x} and the formula shown:(2) Type of real-time and historical control chart found in FactoryLink SPC. See “Sigma,” “EWMA,” etc.

$$\bar{X} = \frac{\sum^x}{n} = \frac{\text{Sum of the values in subgroup}}{\text{Total number of values in subgroup}}$$

- \bar{x} and R Chart For variable data: control charts for the average and range of subgroups of data.
- \bar{x} and Sigma Chart For variable data: control charts for the average and standard deviation (sigma) of subgroups of data.
- XOR Abbreviation for the Exclusive OR, a binary logical operator that yields a value of TRUE (1) if one, but not both, of its two arguments is TRUE (non-zero).
- XREF
(cross-reference)
database table Configuration Manager-maintained database table that contains a record for each occurrence of an element in any task-specific database table or graphics animation.

Index

A

accessing

- help (Run-Time Monitor) 336
- Run-Time Manager 299
- Run-Time Monitor 322

adding

- elements to a watch list 325
- new tasks 281

alarm

- criteria 45
- distribution 57
- grouping 50
- logging 58
- parent/child relationship 51
- persistence 56
- states 49

Alarm Logger, see Distributed Alarm Logger
43

alphanumeric 429

AND 429

animate 429

animation

- creating a graphics display 64

animation object 429

application controls 306

application directory

- files 36
- subdirectories 35

Application Editor

- animating a graphics display 64

creating a graphics display 64

defining tags 205

applications

- converting (FLCONV) 402
- creating new (FLNEW) 362, 392
- demo application (FLDEMO) 396
- exporting 222
- importing 223
- saving (FLSAVE) 403
- test (FLTEST) 396

archiving 429

archiving error messages 286

argument 429

arrays

- defining element arrays 207
- determining next available dimension
282
- maximum number 209
- multi-dimensional 208
- one-dimensional 207

ASCII 429

Assignable Causes (of variation) 429

assignment statement 430

attribute 430

Attribute Data (Quality) 430

Average 447

B

background display 430

background task 430

Bar Graph 449

batch files

- adding remarks 341
- echoing a comment in output 341
- ending current file 342
- executing multiple times 342
- executing once 342
- working with 341

Batch Recipe 430

- creating/animating a graphics display 64
- principles of operation 62

Bell-Shaped Curve 431

binary operator 431

bit 431

blank line, inserting 240

block send 119

boot 432

browse table 432

Browser

- logical expressions 68
- principles of operation 66

byte 433

C

CDBLIST 406

Centerline 435

changing

- font size 245
- tag definitions 235

checking

- databases (DBCHK) 407
- syntax 245

child alarms 52

- child alarm delay 53
- child recovery delay 54
- delay, child not suppressed 53

delay, child suppressed 52

cleaning memory areas (FLSHM) 410

clearing error messages 245

client application 151

client to server data transfer 156

client/server 151

CML

switching from IML 136

using 137

color scheme (Windows) 285

command buttons 218

command line

options (Run-Time Manager) 295

commands

Run-Time Monitor 335

Common Causes 436

compiled mode 136

conditional 436

Configuration Manager

- accessing (OS/2) 213
- accessing (Unix) 213
- accessing (Windows) 212
- changing font size 245
- changing tag definitions 235
- checking syntax 245
- clearing errors 245
- copying a line entry 241
- cutting a line entry 241
- defining element arrays 207
- deleting a line entry 240
- deleting tag definitions 233
- display 215
- editing configuration panels 240
- getting help 243
- inserting blank line 240
- managing text-entry panel files 244

- merging files 244
- multiple configuration tables 225
- multiple development applications 221
- overview 211
- pasting a line entry 242
- predefined elements 210
- reporting 246
- saving file contents 244
- scrolling a window or panel 219
- search for tags 231
- searching for a text string 242
- sharing information between applications 222
- structured configuration panels 217
- tags 201
- text-entry panels 216
- using tags as triggers 210
- viewing current FactoryLink version 238
- viewing field choices 239
- working with tags 226
- configuration mode 436
- configuration panels
 - command buttons 218
 - editing 240
 - scrolling 219
 - structured panels 217
 - text-entry panels 216
- configuration tables
 - definition 436
 - Math and Logic 133
 - multiple 225
- configuring
 - OLE 2.0 Automation Server 142
 - program arguments 199
 - system, See System configuration 273
- constant 436

- Continuous Data 436
- Control (of process) (manufacturing) 437
- Control (of process) (statistical) 437
- Control Limits 437
- control statement 437
- conversations
 - DDE Windows 99
- converting
 - application (FLCONV) 402
 - Math and Logic versions 134
 - tagnames 406
- copying
 - line entry (in configuration panel) 241
- correcting
 - internal errors, Run-Time Manager 320
 - internal errors, Run-Time Monitor 360
- Counters, see Programmable Counters 164
- CP 434
- CPK 434
- Create Link option
 - DDE Windows 104
- creating
 - demo (FLDEMO) 396
 - Math and Logic programs 133
 - new application (FLNEW) 362, 392
 - test application (FLTEST) 396
- CTGEN 406
- CTLIST 406
- cutting a line entry (in a configuration panel) 241
- Cycle 437

D

- DIALOG, see Distributed Alarm Logger 43
- data
 - deleting 83

- grouping 76
 - logging 183
 - ordering 186
- Data Logging
 - overview 27
- data transfer
 - client to server 156
 - server to client 155
- Database Browser task 437
- Database Browser, see Browser 65
- database element 437
- Database Logger
 - See Logger 91
- Database Logging
 - overview 28
- Data-Point Logger 91
 - overview 91
- Data-Point Logging 91
 - Index 94
 - LOGTIME 93
 - vs. Database Logging
 - Database Logging
 - vs. Data-Point Logging 28
- Data-Point Logging Table Schema
 - FLOATVAL 93
 - LARGEINT 93
 - SMALLINT 93
 - TAGDATA 93
- Data-Point Logging Tables
 - ANALOG 92
 - LOGDATA 92
 - LONGANA 92
 - TRENDATA 92
- Data-Point Save file 95
- date, changing 112
- DBCHK 407
- DDE Client
 - DDE Windows 99, 106
 - read operations 106
 - setup (Windows) 108
 - write operations 107
- DDE Server
 - DDE Windows 98
 - setup (Windows) 102
- DDE Windows
 - conversations 99
 - Create Link 104
 - DDE Client 99, 106
 - DDE Server 98
 - establishing a DDE link 103
 - formulas 104
 - messages 101
 - modifying a linked element 105
 - overview 97
 - read operations 106
 - setting up DDE Client 108
 - setting up the DDE Server 102
 - write operations 107
- Deadbanding 179–180
- debugging
 - CDBLIST 406
 - CTLIST 406
- decompressing files (EXPLODE) 408
- default 438
- defining
 - element arrays 207
 - multi-dimensional arrays 208
 - one-dimensional arrays 207
- Defining Element Arrays 207
- deleting
 - data 83
 - elements from a watch list 326

- group data 85
- line entries (from a configuration panel) 240
- tag definitions 233
- development environment 22
- digital 438
- directory 438
- discrete 438
- display controls 307
- display object 438
- Distributed Alarm Logger
 - alarm distribution 57
 - alarm grouping 50
 - alarm states 49
 - child alarm delay 52
 - child recovery delay 53
 - criteria, establishing 45
 - Global Hide tags 55
 - Group Hide tags 55
 - hide alarms 55
 - Individual Hide tags 56
 - logbook 60
 - logging alarms 58
 - methodology 44
 - overview 43
 - parent/child relationship 51
- domain 438, 454, 457
- domain list, viewing 236
- domains 31
 - changing associations 283
 - domain selection 215
 - domain structure 32
 - run-time tasks 33
- dynamic 438
- Dynamic Data Exchange, see DDE Windows and DDE OS/2 97

Dynamic Logging Control panel 96

E

- echoing a comment in output 341
- editing
 - configuration panels 240
 - task information 281
- editing a tag 206
- editor 438
- element arrays
 - defining 207
- elements
 - adding to a watch list 325
 - deleting from a watch list 326
 - finding in a watch list 326
 - viewing cross reference list 229
- ending a batch file 342
- environment variables 38
- error messages
 - clearing 245
 - utilities 412
- Event and Interval Timer, see Timer 109
- event charts 194
 - indexing 81
- Excel 105
- exception data 151
- Exception Logging
 - Filtering 94
- exception send 119
- executing
 - batch file 342
- exiting
 - commands window (Run-Time Monitor) 336
 - Run-Time Monitor 347
- EXPLODE 408

exponential constant 439
 exporting an application 222
 expression 439
 external domain 151

F

FactoryLink
 directory organization 35
 module descriptions 29
 monitoring 331
 version number 238
 FactoryLink Application Restore dialog 393
 FactoryLink ECS Languages dialog 395
 FactoryLink Lite
 configuration guidelines 424
 FALSE 441
 field 441
 field choices, viewing 239
 file 437
 File Manager 113
 overview 113
 file name 441
 Filtering Exception-Logged Tags 94
 finding
 elements in a watch list 326
 flag 441
 FLCONV 402
 FLDEMO 396
 FLKEYVAL 409
 FLLAN
 local station default values 124
 local stations 121
 methodology 118
 monitoring the network 123
 multiple platforms 122
 network groups 122

ordering tag names 129
 overview 117
 receiving data 119
 receiving values from remote stations 120
 remote stations 121
 sending data 119
 sending values to remote stations 119
 supported network operating systems 122
 supported protocols 122

FLNEW 362, 392
 FLOAT 442
 Floating-point 442
 Floating-point constant 442
 FLSAVE 403
 FLSETLNG utility 395
 FLSHM 410
 FLTEST 396
 font 442
 font size, changing 245
 format specifiers 39
 formula 442
 formulas
 DDE Windows 104

G

Global Hide tag 55
 Group Hide tags 55
 grouped by group logging method 86
 grouping data 76
 deleting 85
 indexing 82

H

help
 Configuration Manager 243
 Run-Time Monitor 336

- hide alarms 55
 - Global Hide tag 55
 - Group Hide tags 55
 - Individual Hide tags 56
- Hide tags 55–56
- Histogram 443
- Historian
 - methodology 115
 - overview 27, 115
 - supported relational databases 116
- historical trending 191, 197
- horizontal scroll bar 220

I

- icon 444
- IML 135
- importing an application 223
- indexing for trending
 - event charts 81
 - grouped data 82
 - time-based charts 81
- Individual Hide tags 56
- initialize 444
- input object 444
- inserting
 - blank line 240
- Instability (of a process) 444
- integer constant 444
- integer order 79
- interpreted mode 135

K

- K 445
- KEYINST 409

L

- LAN 445
- LCL
 - Lower Control Limit 445
- licensed options 411
- links
 - DDE Windows 103
- Local Area Network (LAN) 446
- local area networking, see FLLAN 117
- local station default values
 - ACK 127
 - BUFSIZE 127
 - CALL 125
 - INIT 125
 - MAXLEN 126
 - MAXSESS 127
 - RX (Receive Timeout) 124
 - SD (Send Delay) 128
 - ST (Send Timeout) 128
 - TX (Transmit Timeout) 124
- local stations 121, 124
- log 446
- logbook 60
- Logger
 - logging methods 86
 - overview 27
- logging alarms 58
- Logging Data 95
- Logging Data Dynamically 96
- Logging Method
 - change in a trigger tag 94
 - change in the tag 94
 - fixed-time interval 94
- Logging Methods 94
- logical expressions 68
- logical operator 446

LOGTIME 93
 long analog 446
 LONGANA 446
 LSL
 Lower Specification Limit 445

M

Main Menu 215
 Math and Logic
 calculating methodology 132
 calling 139
 compiled mode 136
 compiled procedures 135
 configuration tables 133
 converting previous versions 134
 creating programs 133
 interpreted mode 135
 modes 135
 overview 131
 Procedure table 133
 switching from IML to CML 136
 trigger elements 134
 Triggers table 133
 uses 132
 Variables table 133
 Mean (of a statistical sample) 447
 menu bar 215
 merging files 244
 message 447
 messages
 DDE Windows 101
 MKCML 138
 modifying
 linked element (DDE Windows) 105
 module dependencies 401
 monitoring

FactoryLink 331
 network 123
 real-time database status 343
 monochrome 447
 Moving Average/Moving Range Chart 447
 moving screen components 289
 multi-dimensional arrays 208
 multiplatform save 392, 403
 multiple configuration tables, opening 225
 multiple platforms
 on a network 122
 multiple-user environments 34
 separate applications 34
 shared applications 34

N

NETBIOS 447
 network 447
 operating systems 122
 new application
 opening 221
 node 447
 non-grouped/non-sequenced data
 logging method 86
 non-grouped/sequenced data
 logging method 86
 NOT 448
 np-Chart 448
 numeric constant 448

O

object 448
 OLE 2.0 Automation Server
 configuring 142
 overview 141
 principles of operation 142

- one-dimensional arrays, defining 207
- opening
 - multiple tables 225
 - new development application 221
- operand 448
- Operating Characteristics Curve 448
- Operating System/2 (OS/2) 448
- operator 448
- OR 448
- ordering
 - tag names 129
- ordering data 186
 - integer 79
 - timestamp 79
- OS/2 449
 - accessing Configuration Manager 213
 - accessing Run-Time Manager 299
 - network protocols 122
- output
 - echoing a comment 341

P

- panning 197
- parent/child relationship 51
- Pareto Analysis 449
- Pareto Diagram 449
- PARSECML 138
- pasting
 - line entry (in a configuration panel) 242
- path name 449
- p-Chart (percent defective) 449
- pens
 - pen types 197
- Percent Defective 449
- Persistence
 - configuration changes, resolving 147

- overview 143
- principles of operation 145
- pixel 450
- platform-specific save 392, 403
- PLC 449
- pointing device 450
- poll rate bar 323
- polled data 151
- polled-exception data 151
- polygon 450
- PowerNet
 - client to server data transfer 156
 - data transfer method 155
 - definitions 151
 - methodology 152
 - overview 149
 - server to client transfer 155
 - startup 155
 - tag naming convention 157
 - tag type conversion 159
- Preconfigured Data-Point Logging Tables 92
- predefined elements 210
- Print Spooler
 - overview 161
- Process Analysis Diagram 450
- Process Control 451
- processes
 - monitoring 331
- program arguments
 - Browser 71
 - Trend 199
- Programmable Counters
 - analog and digital values 164
 - elements 164
 - examples 165
 - overview 163

- principles of operation 164
- Programmable Logic Controller 451
- prompt 451

Q

- Quality Characteristic 451

R

- Range 452
- Rational Subgrouping 453
- raw value tag 180
- R-Chart 451
- read operations
 - DDE Client (Windows) 106
- reading
 - element (Run-Time Monitor) 337
 - real-time database elements 329
- real-time 452
- real-time database 24, 452
 - generating data 25
 - monitoring status 343
 - reading data 26
 - reading elements 329
 - viewing elements list 227
 - writing elements 329
- Real-Time Graphics 453
- real-time mode 197
- real-time only trending 190
- reboot 452
- Recipe 452
- record rollover 83
- relational operator 452
- remote station communications
 - receiving values 120
 - remote stations 121
- remote TAG (object) reference 151

Report Generator

- complete triggers 177
- components of a format file 170
- escape sequences 178
- format specifiers 172
- format variations 176
- keywords 170
- location of object names 172
- methodology 168
- overview 167
- placement of reported data 172
- sections of format file 170
- trigger actions 173
- reports
 - Configuration Manager 246
- resizing screen components 289
- RESOLVE 145, 147
- restoring
 - FLDEMO (OS/2) 398
 - FLDEMO (Unix) 400
 - FLDEMO (Windows) 396
 - FLTEST (OS/2) 398
 - FLTEST (Unix) 400
 - FLTEST (Windows) 396
- result table 453
- retrieving
 - watch list 328
- rollover
 - record 83
 - subgroup 84
- run-time 453
- run-time environment 23
- Run-Time Graphics Windows 284
- Run-Time Manager
 - accessing (OS/2) 299
 - accessing (Unix) 300

- accessing (Windows) 299
- application controls 306
- command line options 295
- correcting internal errors 320
- display controls 307
- screen 301, 303
- task controls 305
- user-interface screen 304
- using 293
- Run-Time Mode 453
- Run-Time Monitor
 - accessing 322
 - adding elements to watch list 325
 - correcting internal errors 360
 - deleting elements from a watch list 326
 - exiting 347
 - exiting commands window 336
 - finding elements in a watch list 326
 - help 336
 - monitoring real-time database status 343
 - reading an element 337
 - retrieving watch list 328
 - storing watch list 327
 - terminating a task 336
 - terminating all FactoryLink tasks 346
 - using commands 335
 - viewing element current values 324
 - working with batch files 341
 - writing to an element 339
- S**
 - Sample (Statistics) 453
 - Sample Size 453
 - Sampling 453
 - Sampling Variation 453
 - saving
 - application (FLSAVE) 403
 - saving file contents 244
 - Scaling and Deadbanding
 - overview 179
 - principles of operation 180
 - raw value tag 180
 - scaled value tag 180
 - Schema
 - overview 27
 - schema
 - assigning names 187
 - Schema Overview 183
 - searching
 - tags 231
 - text string 242
 - select object 453
 - server application 151
 - server to client data transfer 155
 - Shared domain 32
 - shared, numeric value tags 91
 - sharing information between applications 222
 - Sigma 454
 - Sigma Limits 454
 - Single Logging Request 96
 - single-threaded task 139
 - solicited data 151
 - Stability (of a process) 454
 - Standard Deviation 454
 - start trigger 454
 - statement 454
 - storing
 - watch list 327
 - string constant 455
 - structure of tag names 202
 - structured configuration panels 217
 - subdirectory 455

- Subgroup 455
- subgroups
 - rollover 84
- supported relational databases 116
- switching from IML to CML 136
- symbol 455
- syntax
 - checking 245
- system configuration
 - adding new tasks 281
 - array dimension, determining 282
 - calculating number of processes 290
 - color scheme, changing 285
 - domain associations, changing 283
 - editing task information 281
 - error messages, archiving 286
 - overview 273
 - Run-Time Graphics Windows 284
 - screen components, modifying 289
 - System Configuration Information panel 274
 - viewing domain associations 281
- system directory subdirectories 36
- system messages
 - archiving 286

T

- tag name 455
 - structure 202
- tag naming convention 157
- tag number 455
- tag parameters 157, 181, 202, 420
- tag type conversion 159
- TAGNAME 93
- tags
 - changing a definition 235

- converting tagnames 406
 - deleting definition 233
 - editing 206
 - parameters 157, 181, 202, 420
 - searching 231
 - using as triggers 210
 - viewing number defined 226
- TAGVALUE 93
- task controls 305
- tasks
 - adding 281
 - terminating 346
- terminating
 - all FactoryLink tasks 346
 - single task 336
- text cursor 456
- text-entry panels
 - managing files 244
 - overview 216
- time, changing 112
- time-based charts
 - indexing 81
 - trending 193
- Timer
 - changing date 112
 - changing time 112
 - overview 109
 - principles of operation 110
- timestamp order 79
- toggle 456
- Tolerance 456
- Trend 456
- Trending
 - chart types 193
 - configuring program arguments 199
 - event charts 194

- historical mode 197
- historical trending 191
- methodology 190
- overview 189
- panning 197
- pen types 197
- real-time mode 197
- real-time only trending 190
- switching modes 197
- time-based charts 193
- value cursor 198
- zooming 197
- trending
 - indexing 81
- trigger elements
 - Math and Logic 134
- triggering schemes 139
- triggers
 - complete triggers 177
 - definition 456
 - trigger actions 173
 - using tags in CM as triggers 210
- TRUE 456
- type 456

U

- u-Chart 456
- UCL
 - Upper Control Limit 456
- UKEY 411
- unary operator 457
- UNIX
 - network protocols 123
- Unix
 - accessing Configuration Manager 213
 - accessing Run-Time Manager 300

- unsolicited data 151
- User domain 33
- user-interface screen 304
- using format specifiers 39
- USL
 - Upper Specifications Limit 456
- utilities
 - CDBLIST 406
 - CTGEN 406
 - CTLIST 406
 - DBCHK 407
 - EXPLODE 408
 - FLCONV 402
 - FLDEMO 396
 - FLKEYVAL 409
 - FLNEW 362, 392
 - FLSAVE 403
 - FLSHM 410
 - FLTEST 396
 - KEYINST 409
 - messages 412
 - UKEY 411

V

- value cursor
 - definition 198
- Variable Data 457
- Variables 457
- Variance 457
- version
 - viewing current 238
- vertical scroll bar 219
- viewing 411
 - authorized options (UKEY) 411
 - current FactoryLink version 238
 - database elements 227

- domain associations 281
- domain list 236
- element cross reference 229
- element values (Run-Time Monitor) 324
- field choices 239
- number of tags defined 226

W

- watch list
 - adding elements 325
 - deleting elements 326
 - finding elements 326
 - retrieving 328
 - storing 327
- window 457
- Windows
 - accessing Run-Time Manager 299
 - color scheme 285
 - network protocols 122
- write operations
 - DDE Client (Windows) 107
- writing
 - real-time database elements 329
 - to an element 339

X

- x and R Chart 458
- x and Sigma Chart 458
- XOR 458

Z

- zooming 197