



iFIX 6.1

Building a SCADA System

Proprietary Notice

The information contained in this publication is believed to be accurate and reliable. However, General Electric Company assumes no responsibilities for any errors, omissions or inaccuracies. Information contained in the publication is subject to change without notice.

No part of this publication may be reproduced in any form, or stored in a database or retrieval system, or transmitted or distributed in any form by any means, electronic, mechanical photocopying, recording or otherwise, without the prior written permission of General Electric Company. Information contained herein is subject to change without notice.

© 2020, General Electric Company. All rights reserved.

Trademark Notices

GE, the GE Monogram, and Predix are either registered trademarks or trademarks of General Electric Company.

Microsoft® is a registered trademark of Microsoft Corporation, in the United States and/or other countries.

All other trademarks are the property of their respective owners.

We want to hear from you. If you have any comments, questions, or suggestions about our documentation, send them to the following email address:

doc@ge.com

Table of Contents

Building a SCADA System	1
Reference Documents	1
Introduction	2
Database Manager	2
Understanding a Database	2
Understanding Database Blocks	3
Understanding Chains	3
Processing the Database	4
Using the Process Database	5
Trending Process Data	5
Creating Scripts and Schedules with Process Data	5
Archiving Process Data	5
Sample Application	6
Getting Started	7
To get started:	7
Starting and Stopping Database Manager	7
Using the Database Spreadsheet	7
Understanding Spreadsheet Properties	7
Working with the Pop-up Menu	8
Editing the Spreadsheet	8
Exchanging Data	8
Working with the Database Manager Ribbon	8
Setting Database Manager Preferences	9
Creating a Process Database: Overview	9
To create a process database:	9
Implementing a Process Database	11
Sample Process Application	11
Designing a Chain	12
To design a chain:	12

Describing the Sample Process Application	12
Analyzing the Sample Process	13
Designing an Automation Strategy for the Sample Process	14
Writing an Automation Algorithm for the Sample Process	14
Matching Process Steps to Block Types	15
Drawing a Flowchart for the Sample Chain	16
Understanding Chain Designs	20
Building Large Databases	20
Saving Large Databases	21
Working with Analog and Digital Blocks	21
Working with Database Blocks	21
Block Types and Descriptions	22
Understanding Primary Blocks	23
Understanding Secondary Blocks	24
Understanding Control Blocks	25
Understanding Statistical Process Control Blocks	25
Understanding Batch Blocks	25
Understanding SQL Blocks	26
Understanding Database Dynamos	26
Database Dynamo Configuration Utility (BTKCFG.exe)	27
To add a Database Dynamo:	27
To remove a Database Dynamo:	28
To change a slot number of a Database Dynamo:	28
Understanding Slot Numbers	28
Block Fields	28
Locating Block Fields	29
Completing Block Dialog Boxes	29
Scan Times	29
Understanding Time-Based Processing	30
Assigning Time-Based Scan Times	30

Example: Assigning Time-Based Scan Times	30
Understanding Exception-Based Processing	31
Blocks that Work Best with Exception-Based Processing	31
Using Program and Statistical Data Blocks	32
Blocks that Receive Multiple Inputs	32
Blocks that Define a Time Constant	32
Using Analog Alarm and Digital Alarm Blocks	32
Assigning Exception-Based Scan Times	33
Understanding One Shot Processing	33
Phasing	33
Phasing Second and Subsecond Blocks	34
Assigning Phase Times	34
Overphasing	35
Overphasing and Long Scan Times	35
Block Modes	35
Understanding PMAN Mode	37
Example: Blocks in PMAN Mode	37
Understanding Blocks in PAUT Mode	37
Example: Blocks in PAUT Mode	37
Placing Blocks On and Off Scan	38
Blocks with Long Scan Times	39
Example: Placing Blocks with Long Scan Times On Scan	39
Using the S Parameter	39
I/O Drivers	39
OPC Client Driver	39
Simulation Drivers	40
Using the OPC Client I/O Driver	41
To use the OPC I/O Driver:	41
Using the Simulation Driver	41
To use the SIM driver:	41

Examples: SIM Addresses	42
Using Signal Generation Registers in the SIM Driver	43
To assign one of these registers to a block:	43
Example	43
Using the Simulation 2 Driver	44
Accessing SM2 Registers	44
To use the SM2 register:	44
Generating Bad Data with the SM2 Driver	45
Using the SM2 C API	45
Example	45
Understanding Alarm Statuses	48
Connecting to an OPC Server	49
Understanding Signal Conditioning	49
Example: Understanding Signal Conditioning	50
Understanding EGU Limits	50
Changing the EGU Limit Precision	51
Changing the EGU Limit Range	51
EGU Limit Formats	51
Working With the Process Database	53
Creating a New Database	53
Opening and Closing a Database	53
Adding Blocks	53
Adding Multiple Blocks	54
Duplicating Blocks	56
Copying and Pasting Blocks	56
To configure Excel:	56
Moving Blocks	57
Modifying Blocks	58
Displaying Block Information	58
Deleting Blocks	59

Saving a Database	59
Saving a Database from a Script or a Program Block	59
Locating and Displaying Data	60
Finding Data in a Spreadsheet	60
Find Options	60
Replacing Data	61
Find Options	61
Using Go To	61
Updating and Pausing the Spreadsheet	61
Customizing the Spreadsheet	62
Coloring the Spreadsheet	62
Creating a Non-Scrolling Column	62
Defining a Spreadsheet Layout	63
Saving and Loading Spreadsheet Layouts	64
Overriding the Default Layout	64
Troubleshooting Tag Display	65
To resolve tag display problems:	65
Querying and Sorting Data	66
Understanding Query Syntax	66
Case Sensitivity	67
Using Boolean Operators	67
Using Wildcards in a Query	67
Examples: Query Wildcards	68
Examples: Using Relational Operators	68
Grouping Queries	68
Editing a Query	68
To locate blank cells in the database:	69
Refining and Expanding a Query	69
Saving and Loading a Query	69
Overriding the Default Query	69

Understanding Sort Orders	70
How Sort Orders Work	70
Saving and Loading a Sort Order	70
Changing the Default Sort Order	71
Managing Databases	71
Verifying Databases	71
Correcting Errors	72
To correct errors:	72
Reloading Databases	72
Reloading a Database from a Visual Basic Script	73
Displaying a Database Summary	73
Exporting Databases	74
Exporting a Database from the Command Line	74
Syntax	75
Parameters	75
Error Codes	75
Editing the Export File	75
CSV File Format	76
GDB File Format	76
Importing Databases	77
To change the scanning order of a database's blocks:	77
To combine two process databases:	78
Advanced Topics	78
Changing a Database's Scanning Order	78
Understanding a Database's Scanning Order	79
Example: Understanding a Database's Scanning Order	79
Changing the Order of Solve	80
Customizing the Toolbar	81
Customizing the Tools Menu	81
Index	82

Building a SCADA System

Building a SCADA System is intended for process engineers responsible for designing and building a process database. This manual shows engineers how to create, modify, and delete blocks and chains. It also teaches the skills engineers need to sort, query, and optimize iFIX® databases.

Reference Documents

For related information about iFIX, refer to the following documents:

- [Understanding iFIX](#)
- [Writing Scripts](#)
- [Creating Recipes](#)

Introduction

iFIX® provides process information for plant managers, supervisors, and operators in the form of reports, displays, archived data, alarms, messages, and statistical charts. The sources of this information are OPC servers or process hardware – the controllers, sensors, motors, switches, and other devices – required to manufacture your product.

iFIX reads process information from these devices and saves it in one or more *process* databases residing on your SCADA servers. The database plays an integral part in your industrial automation strategy; it is the primary source of process data for most iFIX applications. Whether you collect historical values or generate shift reports, iFIX enables you to create a database that supports your specific industrial control and automation needs.

Database Manager

Your main tool for creating and managing process databases is Database Manager. This program lets you open and configure the database of any SCADA server. You can also:

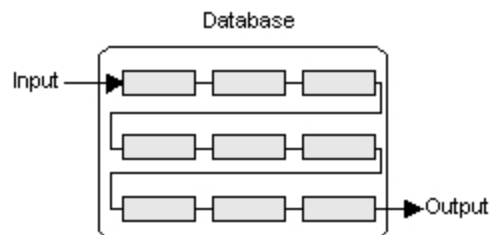
- Query and sort the database.
- Find and replace database information.
- Import and export a database.
- Generate multiple database blocks automatically.
- Customize your display.

Understanding a Database

Every SCADA server loads a process database at start-up. Once loaded, the database:

- Receives values from an I/O driver or OLE for Process Control (OPC) server.
- Manipulates values according to its configuration (a control strategy).
- Compares the values to alarm limits you define.
- Outputs adjusted values to the I/O driver or OPC server.
- Sends alarms to operator displays, printers, files, and network alarm destinations.

The following figure illustrates how values enter a database, travel through a sequence of blocks, and exit the database as output.



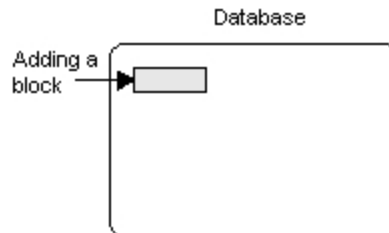
Understanding Database Blocks

The main components of a process database are blocks. Blocks are capable of:

- Receiving values either from another block, OPC server, or directly from an I/O driver.
- Manipulating values according to its configuration.
- Comparing incoming values against pre-defined limits.
- Scaling process values to a specified range.
- Performing calculations.
- Outputting values back to the I/O driver or OPC server.

iFIX provides different types of blocks, each capable of performing a unique function. For a brief description of each block, refer to the section [Block Types and Descriptions](#).

By default, when you install iFIX, it creates an empty database for your SCADA server. Using Database Manager, you can add the blocks you require to this database.



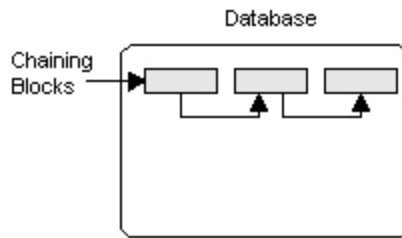
Next, you configure the block by entering:

- The block's name.
- How it receives data from an I/O driver, OPC server, or from another block.
- Where it sends information.
- If it manipulates values.
- How it reacts to critical value changes (called *alarming*).
- How it scales values for operator displays.

Understanding Chains

Blocks by themselves perform specific tasks in the database. By combining two or more blocks together, you can form *chains*. Each chain performs the tasks of its component blocks by passing data from one block to the next. When properly configured, chains can generate alarms, acquire data, and verify, automate, and maintain a process. In automating large processes

involving a number of I/O devices, a database can contain many chains, each designed to automate and maintain a specific function or process step.



Each chain can contain up to 30 blocks, with each block configured to perform a specific processing function. Note that some blocks are designed to work in chains while others are designed to operate on their own. Specific types of blocks, their functions, and their relationship to one another are summarized in the section [Block Types and Descriptions](#).

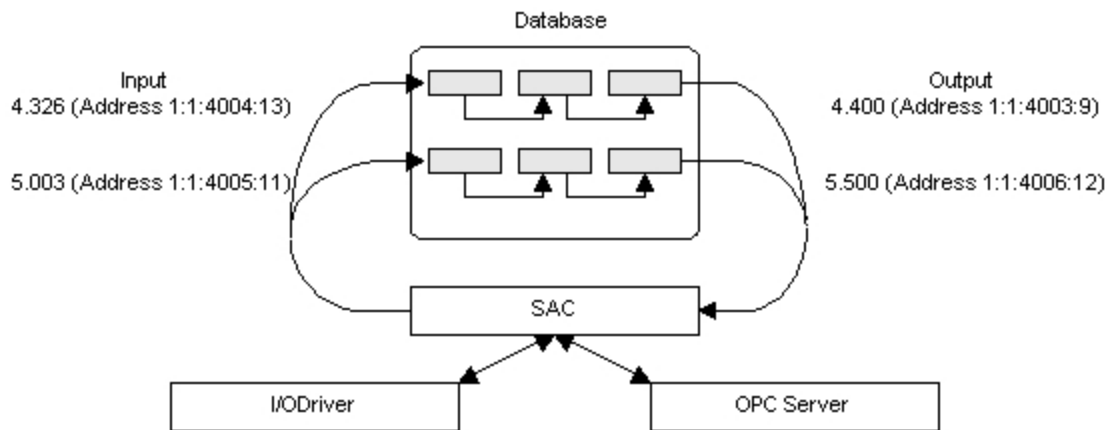
Familiarize yourself with the available blocks. Having a good understanding of the blocks and their capabilities allows you to quickly design optimum strategies for your application.

Processing the Database

In order for your blocks and chains to function, the Scan, Alarm, and Control program (SAC) processes them at the time interval you define. SAC processes your blocks and chains by:

- **Scanning** individual blocks in the chain, reading new I/O values and returning values to the process hardware.
- **Alarming** on incoming values if they exceed pre-defined limits you specify.
- **Controlling** the automation process by assuring that each block verifies or manipulates incoming values according to its configuration and sends values on to the next block in the chain.

The following figure shows SAC reading process values, sending these values through database chains, and returning the output values.



SAC Processing I/O Data

Using the Process Database

Once you create a process database, your SCADA server can monitor process conditions and report alarms. Alarms occur when an incoming value exceeds a pre-defined limit. Using the iFIX WorkSpace, you can draw and animate any object by rotating, coloring, or scaling it when an alarm occurs. These visual cues help operators respond to alarms in a timely fashion. For more information on reporting alarms, refer to the [Implementing Alarms and Messages](#) manual.

For information on trending data, creating scripts and schedules, and archiving data, refer to the following topics:

- [Trending Process Data](#)
- [Creating Scripts and Schedules with Process Data](#)
- [Archiving Process Data](#)

Trending Process Data

SCADA servers can also trend and display real-time and historical values. Process trends enable you to record and analyze process-critical values and allow you to:

- Archive process variables to meet federal regulations.
- Monitor the efficiency of products.
- Maintain equipment.
- Analyze post-process data.

For more information on collecting historical values, refer to the [Trending Historical Data](#) manual. For more information on trending real-time data for operators, refer to the [Creating Pictures](#) manual.

Creating Scripts and Schedules with Process Data

Using Visual Basic for Applications (VBA) or the Scheduler, you can adjust database values, place blocks on scan or off scan, change a block's mode, or launch a script based on a value change or an alarm. For example, using a time-based schedule, you could set the values of a dozen blocks prior to the manufacture of your product or you could automatically ramp a value based on an event (such as a valve closing). Likewise, you could write a script to monitor the value of a block and, when an alarm occurs, open a diagnostic picture. For more information on using these features, refer to the [Writing Scripts](#) and [Mastering iFIX](#) manuals.

Archiving Process Data

In addition to using scripts and schedules to write values to the process database, you can use a relational database. Writing values from a relational database is similar to writing values with a script. In both cases, values are written at a pre-defined time or when an event occurs (such as a contact opening). However, unlike scripts and schedules, writing data from a relational database requires that you configure the SQL Trigger and SQL Data blocks in the process database.

These blocks also let you archive data to a relational database. Once the relational database receives and stores the process data, you can query it to retrieve and analyze any information. For more information on setting up and using a relational database, refer to your ODBC manuals and the [Using SQL](#). For more information on process database blocks, refer to the [iFIX Database Reference](#).

IMPORTANT: If you are working in a secure environment and using the Electronic Signature option, you must be aware of the impact of unsigned writes to the process database. Unsigned writes can originate from scripts, schedules, and writes from a relational database. Refer to the [Implications of Database Writes With Electronic Signature](#) section of the Using Electronic Signatures manual for detailed information.

Sample Application

To help you understand how to create process databases for yourself, this manual uses a sample application for manufacturing fertilizer and monitoring chemical consumption and equipment usage. The blocks and chains that comprise the application were developed by the Enviro company which collects sludge generated by municipal sewage plants and converts it into different grades of agricultural fertilizer in briquette form. The Enviro company's manufacturing process encompasses:

- Continuous control
- Batch control
- Discrete parts

Getting Started

You can begin creating process databases by verifying that your SCADA servers are set up and functioning. Once you set up these computers, follow the steps.

► To get started:

1. Install iFIX on each SCADA server.
2. Configure each server with the System Configuration Utility (SCU). Refer to the [Setting up the Environment](#) manual to learn how to enable SCADA support.
3. Set up the alarm area database for each server. This database contains the alarm areas you can assign to the blocks on the SCADA server. If you want to share an alarm area database across multiple servers, change the Alarm Area path in the SCU to reference a file server before you create your process databases.

Starting and Stopping Database Manager

You can start Database Manager by clicking the Database Manager button on the Application toolbar (Classic view) or by selecting Database Manager, in the Process Database group, on the Applications tab (Ribbon view). As the program starts up, it prompts you to select the SCADA server you want to connect to and establishes a connection to the computer you select. Once Database Manager connects to the selected SCADA server, the program opens the server's current database.

From the Database Manager, you can open databases from SCADA nodes with iFIX 3.0 or greater installed. You cannot add, modify, or delete blocks in databases from earlier versions of iFIX, such as iFIX 2.5 or FIX32.

To exit from the program, in Classic view, select Exit from the File menu or in Ribbon view, click the Database Manager button, and then click Exit.

NOTE: Connections to remote SCADA servers are established with a physical node name, not logical node names. Logical and physical node names are identical unless you enable SCADA redundancy. For more information about this feature, refer to the [Mastering iFIX](#) manual.

Using the Database Spreadsheet

Each process database you open appears as a spreadsheet in Database Manager. Blocks appear as rows, and block fields appear as columns. Click to select a row or field. You can also select multiple rows and columns by clicking and dragging.

The following topics provide more information about the database spreadsheet:

- [Understanding Spreadsheet Properties](#)
- [Working with the Pop-up Menu](#)
- [Editing the Spreadsheet](#)

Understanding Spreadsheet Properties

The database spreadsheet has many properties you can configure. Among these are:

- The sort order.
- The default query.
- The display format.
- The color scheme.
- The font properties.

By configuring these properties you can customize the spreadsheet as needed. To learn more about these properties, refer to the chapters [Locating and Displaying Data](#) and [Querying and Sorting Data](#).

Working with the Pop-up Menu

In addition to the properties you can configure, the spreadsheet lets you right-click any row, column, or cell to display a menu containing a list of frequently-accessed commands. These commands are identical to commands you can access from the menu bar. For example, Cut on the pop-up menu is identical to Cut on the Edit menu. Similarly, Add Block on the pop-up menu is identical to Add on the Blocks menu.

Editing the Spreadsheet

You can edit most spreadsheet cells by selecting them and entering the data you want to display, allowing you to change individual items quickly. Cells in certain columns, such as the Tagname column, are read-only and cannot be modified.

To change multiple fields for a particular block, you can double-click the block to display its dialog box. To change the same field for multiple blocks, you can find and replace data in a spreadsheet column. Refer to the chapter [Locating and Displaying Data](#) to learn more about finding and replacing data.

Exchanging Data

Database Manager also lets you edit blocks by copying or exporting them. Whenever you complete these tasks, the program converts the selected blocks into comma separated value (CSV) format, enabling you to paste or import the data into your favorite text editor or spreadsheet. Once you edit the data, you can copy or import the data back to Database Manager.

Often a quicker way to convert blocks into CSV format is by holding down the Control key and dragging and dropping the blocks from Database Manager and into Microsoft® Excel. The blocks you move appear in CSV format, allowing you to edit them. When you finish, you can drag them back. To learn more about using Excel with the Database Manager, refer to the section [Copying and Pasting Blocks](#).

Working with the Database Manager Ribbon

Across the top of the spreadsheet is the Database Manager Ribbon. The Ribbon, shown in the following figure, provides access to common database operations such as saving a database or adding a block.



Setting Database Manager Preferences

Database Manager provides preferences for printing, saving, displaying, and editing the database. These preferences let you adjust how Database Manager handles certain operations. The following table summarizes the preferences you can enable.

Database Manager Preferences

The...	Let you...
Printing preferences	Print the database while you configure a block or start another iFIX program.
Saving preferences	Save the current spreadsheet settings or the process database when you exit Database Manager.
Display preferences	<p>View or hide:</p> <ul style="list-style-type: none"> • Spreadsheet row numbers • The status bar • The Database Manager toolbar <p>You also have the option to set the automatic refresh rate.</p>
Editing preferences	<p>Open block dialog boxes in view or modify mode when you double-click a cell in the spreadsheet. In view mode, you can examine the block configuration but not modify it. The block and its chain remain on scan at all times. In modify mode, you can examine and change the block configuration. The block and its chain are placed off scan even if you do not modify the block.</p> <p>You also have the option to place edited blocks and chains on scan automatically when you finish editing them. By default, this option is disabled and Database Manager prompts you to place modified blocks on scan.</p>

Creating a Process Database: Overview

You can create a process database by following these steps.

► To create a process database:

1. Create primary blocks for your I/O points.
2. Create any additional blocks you need and connect them to your primary blocks to form chains.
3. Verify the database for errors.
4. Correct any errors and fine-tune the database.

To learn how to create database blocks, refer to the section [Adding Blocks](#). For information on using standard VBA naming conventions when creating tags, refer to [VBA Naming Conventions](#). For information on verifying and correcting database errors, refer to the section [Verifying Databases](#).

Implementing a Process Database

Implementing a SCADA strategy requires you to collect the following information:

- Flowcharts of your processes.
- A list of I/O driver or OPC server addresses.
- Your alarm requirements.
- The type of signal conditioning and the range of values that your process hardware can measure. The range of values is called the *EGU range*.

After you have this data, you are ready to design a database. To help illustrate how to do this, this chapter walks you through the process of designing a chain for the Enviro company, introduced in the section [Sample Application](#). Using these examples, you should be able to create your own blocks and chains for your process database.

As you read this chapter, you may find it helpful to skip ahead and learn how to add a block with Database Manager so that you can create the sample chains described here. By creating sample chains, working with Database Manager, and correcting errors, you learn how to avoid mistakes before you create your actual database.

Sample Process Application

The first step in designing a database is to examine the application that you want to automate. Next, assign specific processing, maintenance, and monitoring tasks. For example, the Enviro company identified the following tasks, the specific iFIX features they would use to accomplish them, and the economic benefits.

The key elements of this application also apply to processes in other industries, such as chemical processing, food processing, and even discrete parts manufacturing.

Automating the Sample Process with iFIX

Production Tasks:	iFIX Application:	Economic Benefits:
<ul style="list-style-type: none">• Schedule production with sales orders.• Process sludge into fertilizer.• Produce different grades of fertilizer on demand.• Monitor chemical usage and maintain inventory.• Maintain equipment.	<div>Schedule recipes for batch processing.</div> <ul style="list-style-type: none">• Perform supervisory control, direct digital control, and alarming.• Download a recipe for each grade of fertilizer.• Trend chemical levels and print chemical usage reports.• Trend pump/value usage and print maintenance reports.• Historically trend the process.	<div>Increase production through scheduling.</div> <ul style="list-style-type: none">• Produce a higher quality product.• Increase plant availability by scheduling preventative maintenance.• Lower operating costs by making better use of materials and lowering labor cost.• Increase responsiveness to market conditions by varying product size, compound, color, and water content.• Increase profit.

- Maintain fertilizer count.
- Archive processing data.
- Calculate the total product weight according to batch and perform statistical quality control.

Designing a Chain

Once you know the tasks you want to accomplish, you can design the chains for your database. The easiest and most efficient way to design a functional chain is to use the following design steps:

► To design a chain:

1. Analyze your process and make a record of the data you need, including:
 - I/O addresses of the device controller (a wiring list)
 - Device types
 - Signal conditioning
 - The EGU range
2. Design an automation strategy that explains how you want your process automated. For example:
 - What types of alarms and alarm limits do you want to establish?
 - How often do you want the chain processed?
 - When do you want operators notified of process events?
 - How do you want process disruptions handled?
3. Create an algorithm that combines your process analysis and automation strategy. This determines how iFIX automates your process and provides the specific processing instructions you will enter into block dialog boxes.
4. Match the steps in your algorithm to blocks capable of performing these steps.
5. Draw a flowchart listing the block types that perform each step along with the specific processing instructions for each block.

The example presented in the following subsections illustrates the thought process involved in designing a simple chain.

- [Describing the Sample Process Application](#)
- [Analyzing the Sample Process](#)
- [Designing an Automation Strategy for the Sample Process](#)
- [Writing an Automation Algorithm for the Sample Process](#)
- [Matching Process Steps to Block Types](#)
- [Drawing a Flowchart for the Sample Chain](#)

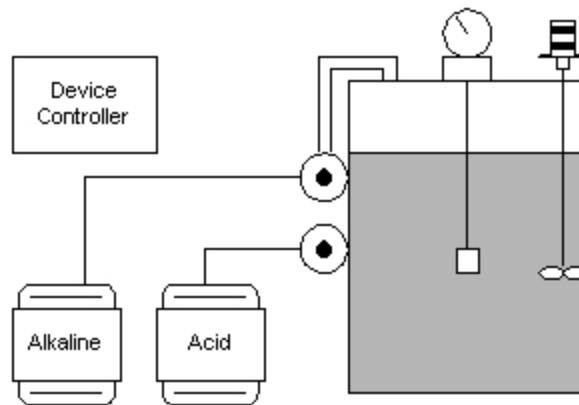
Describing the Sample Process Application

One step in the Enviro company's sludge conversion process is neutralizing acids and alkali in the sludge. The treatment process involves four steps:

1. Adding and mixing water with the sludge.
2. Adjusting the sludge's pH to neutralize any acid or alkali.
3. Adding potassium, nitrogen, and phosphorus to produce different grades of fertilizer.
4. Pumping the prepared sludge to the filter press where excess water is removed to form fertilizer briquettes.

The remainder of this chapter focuses on the second step: pH adjustment.

Sludge from the collection tank is periodically pumped into a large tank where water is added to make the sludge easier to mix. A sensor in the tank registers the sludge's pH level. If the sludge is acidic, a device controller turns on a pump that adds an alkaline solution into the tank to raise the pH; if the sludge is alkaline, the device controller turns on a pump that adds an acidic solution into the tank to lower the pH. In either case, the sludge's pH is properly adjusted according to the grade of fertilizer that is needed before it is pumped to the filter press. The following figure shows the pH adjustment tank and its equipment.



pH Adjustment Tank

The next step is to design a chain that examines and adjusts the sludge's pH by:

- Retrieving information from the pH sensor (data acquisition).
- Sequencing pumps and maintaining pH (Direct digital control).

Once you complete these tasks, you can add blocks to monitor pH fluctuations, chemical usage, pump usage, down time, and other statistical information. You can also historically trend this information, print it in reports, and display it to operators.

Analyzing the Sample Process

The first step in automating your process is analyzing how the process works. By developing a detailed process analysis first, you save time because you isolate the tasks that simple chains can complete and identify those tasks that require more complex chain designs. For example, a task in the Enviro company might be "if the pH is greater than 8.5, add the acidic solution until it reaches 7.5."

The steps required to adjust the pH are:

1. Test the pH each hour.
2. If the pH is below 5.5, gradually add an alkali solution; if above 8.5, gradually add an acidic solution.
3. Stop adding the solution when the pH is between 6.5 and 7.5.

Although this analysis describes how the process works, it lacks important information about your control devices. You need to add detailed information on I/O driver or OPC server addresses, the type of controller equipment, signal conditioning, and maximum ranges the equipment can accept. In short, record all detailed information on how the process equipment operates.

After you obtain this information, you can go on to write a detailed analysis of your process. The following table presents a sample analysis of the pH monitoring and adjustment process.

Sample Process Analysis	
The pH sensor analog address:	1:1:30001
Acid pump's digital address:	1:1:40004:8
Alkali pump's digital address:	1:1:40004:5
Device controller:	MB1
Signal conditioning:	LIN
EGU range:	0.0 to 14.0

Designing an Automation Strategy for the Sample Process

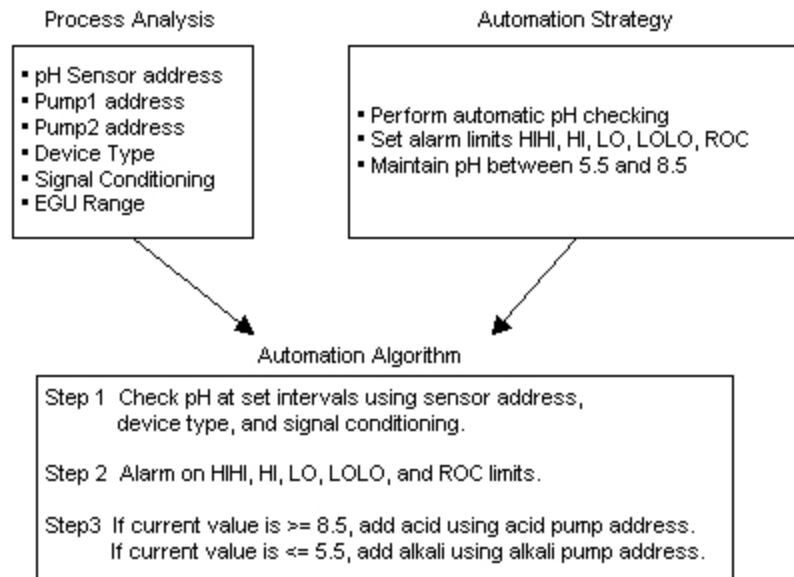
The next step in designing a database chain is planning a successful automation strategy. This requires you to determine how you want to automate the steps recorded in your process analysis. The following steps provide a sample automation strategy for the pH adjustment process.

1. Receive pH information from the sensor at set intervals.
2. Perform alarming. Set HIHI, HI, LO, LOLO, and ROC (rate of change) alarm limits.
3. Sequence pumps to maintain pH.
4. If the pH is below 5.5, turn on the alkaline pump.
5. If the pH is above 8.5, turn on the acid pump.
6. Repeat step 1 through 5 at set intervals.

Notice how this automation strategy incorporates an alarming plan for handling process upsets, such as extreme pH fluctuations (rate of change). This helps to assure that empty drums of acidic and alkaline solutions do not go unnoticed and equipment failures do not interrupt your treatment process.

Writing an Automation Algorithm for the Sample Process

The next and most important step in designing the pH adjustment chain is writing an algorithm that combines both the process analysis and automation strategy. The following figure shows how the technical specifications outlined in the process analysis are combined with the automation needs listed in the automation strategy.

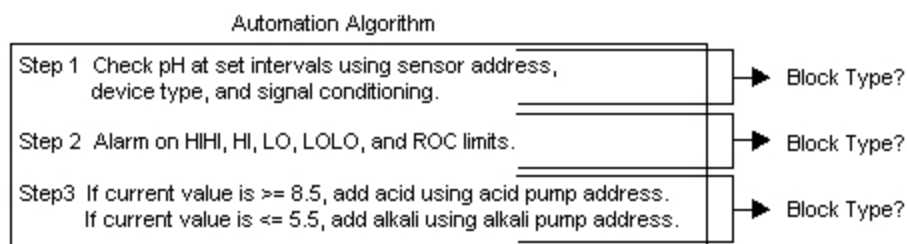


Creating an Automation Algorithm

The algorithm forms the basis of your chain; it identifies the order in which the process proceeds, where each step acquires information, and how iFIX gathers, processes, and verifies information.

Matching Process Steps to Block Types

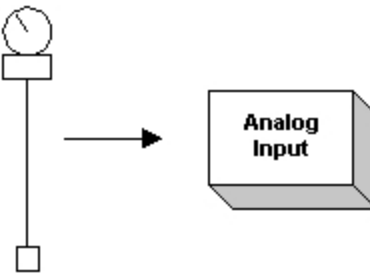
The next step is finding the right database blocks to perform the steps outlined in the algorithm. This process transforms the algorithm from written steps into a chain schematic. Moreover, this step ensures that you incorporate iFIX capabilities into your processing plans. Once you complete this design step, you can enter your designs into your SCADA server's database.



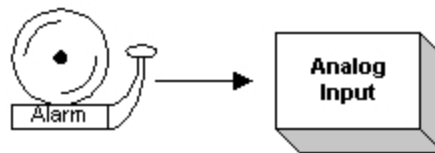
When matching blocks to steps, select a block that can perform the function of a step. In many cases, one block type can handle more than one step, whereas some steps may include too many functions or too many complex operations for any one block.

For your pH adjustment process, the following blocks satisfy the algorithm:

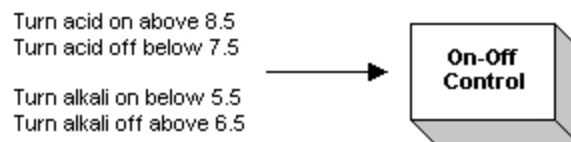
1. An Analog Input block can receive analog signals from the pH sensor's address.



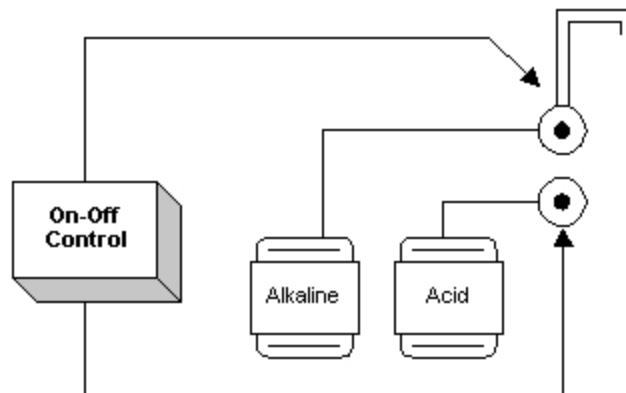
2. The Analog Input block can also accommodate the alarming requirements, so this task is assigned to this block.



3. The pumps are turned on or off based upon the value that the Analog Input block receives from the pH sensor. The block required for this step must be able to monitor an analog input and, depending on the value of the input, open or close a digital output point.



The digital outputs are sent directly to the acid and alkali pump addresses. The On-Off Control block can perform this function.



Since the On-Off Control block can turn the pumps on and off without the use of any other block, this completes step 3.

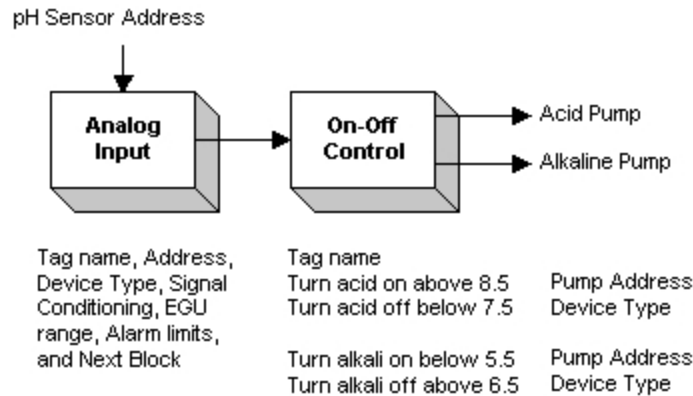
At this point you've identified the blocks that can perform all the steps in your algorithm.

Drawing a Flowchart for the Sample Chain

The last step is to draw a flowchart showing:

- The chained blocks.
- Their names.
- Their I/O addresses.
- Additional instructions on how these blocks handle information.

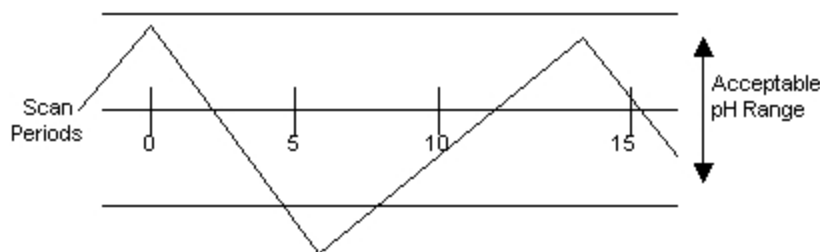
The flowchart is a visual representation of the chain and shows the instructions you will enter into each block as you add them to the database. The following figure shows a flowchart for the pH adjustment process.



Chain Flowchart

Typically, your chain designs are seldom perfect the first time you implement them. With Database Manager, you can always re-evaluate the design of your chain, add blocks, remove blocks, modify block configurations, or change block types to those that accommodate more sophisticated automation functions.

The best approach to re-evaluating the design of your chain is to examine the original automation algorithm and determine if other blocks provide more flexibility or features over the existing blocks. For example, after testing the sample chain, Enviro engineers found it works well most of the time, but every so often the adjustment process exceeds acceptable pH limits by pumping in too much acidic solution, which lowers the pH beyond acceptable limits. The following figure illustrates this problem.



Sample pH Problem

You want the capability of pumping in acid at short intervals. In the original design, iFIX scans the chain every four minutes. This allows the acid more time to react with the sludge *before* more acid enters the tank. An additional improvement would be to shorten the chain's scan time, providing more pH sampling to determine which solution needs to be pumped into the tank.

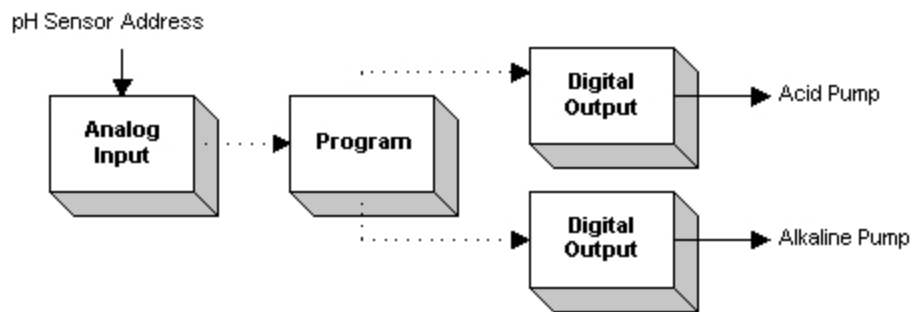
By re-examining the automation algorithm, you can determine where to make improvements to the chain. Reproduced below is the original automation algorithm that called for the On-Off Control block.

- If the pH is below 5.5, turn on the pump to add an alkaline solution. If the pH is above 8.5, turn on the pump to add an acid solution.
- Acid pump digital address: 1:1:40004:8
- Alkaline pump digital address: 1:1:40004:5
- If the pH is between 5.5 and 8.5, stop all pumps.

What you want to do is modify this algorithm to run the acid pump for shorter intervals, giving the acid time to decrease the pH before the acid is turned on again. To do this, you need to examine if other blocks can perform this task.

One block capable of turning a pump on or off is the Program block. You can replace the On-Off Control block with the Program block. However, the Program block cannot establish direct digital contact with the acid and alkali pumps. For this you need to use two Digital Output blocks, one for each pump.

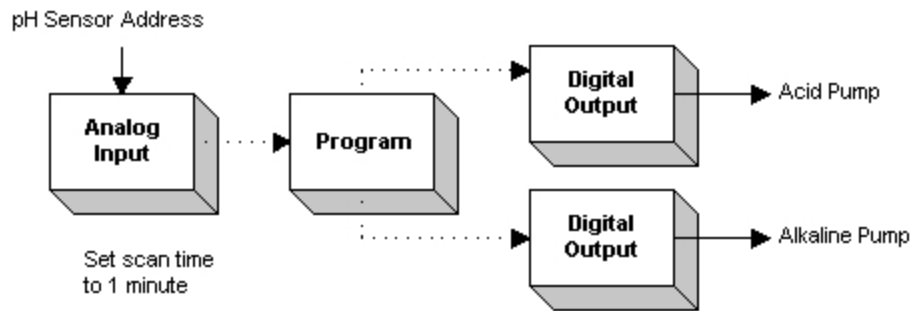
The following figure shows the replacement of the On-Off Control block with the Program block and the two Digital Output blocks, and represents how information passes from one block to the next.



Chain Modification

NOTE: The figure Chain Modification illustrates how the data flows from one block to another, not the actual chain structure.

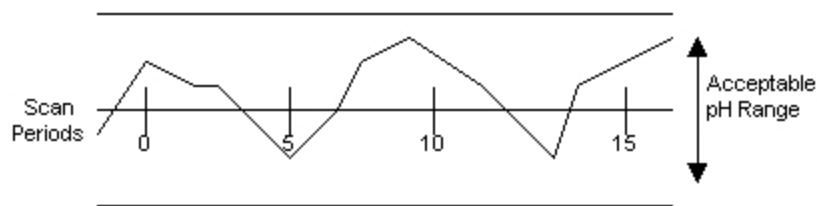
The following figure provides the new programming instructions required to operate the Program and Digital Output blocks.



If pH is ≤ 5.5 , turn alkali pump on
 Wait 1 minute
 Check pH: If pH is ≤ 6.5 , repeat first step
 else continue
 Turn pump off
 If pH is ≥ 8.5 , turn acid pump on
 Wait 1 minute
 Check pH: If pH is ≥ 7.0 , repeat first step
 else continue
 Turn pump off

Modified Flowchart

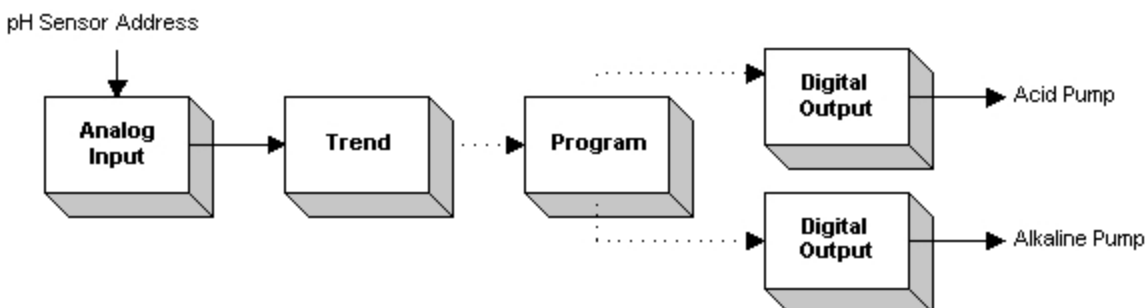
With this modification and with the Analog Input block's scan time reduced to one minute, the chain handles pH adjustment more efficiently by allowing the Program block to run short programs that adjust the pH. The following figure shows how the modified chain handles pH adjustment.



Proper pH Adjustment Trend

With the pH adjustment complete, you might want to consider how to provide more control over the monitoring process. For example, if your operators need to monitor pH fluctuations, you can use a chart to provide a real-time trend for operators.

Data trended by a chart is not stored. To store the real-time data use a Trend block. Then with a chart you can view these values directly on an operator display. The following figure shows the same control chain with a Trend block added to the chain.

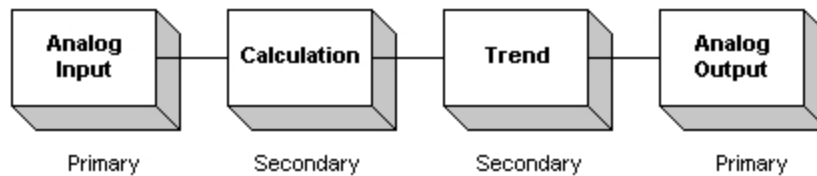


Trend Block Insertion

Adding other blocks, such as the Statistical Data block, allows you to display statistical data in a graph.

Understanding Chain Designs

The chain shown from the sample application performs data acquisition and control. These types of chains usually receive inputs from an I/O driver or OPC server through an input block, manipulate the inputs with secondary blocks, and return an adjusted value to the I/O driver or OPC server through an Analog Output, Digital Output, or On-Off Control block, as the following figure shows.



Sample Data Acquisition and Control Chain

You can also create chains that collect data for monitoring purposes. These types of chains usually receive inputs from an I/O driver or OPC server through an input block but may not return the inputs, since no process adjustment is required. The following figure shows a sample monitoring chain design.



Sample Monitoring Chain

The number of chains you can construct is limited only by memory.

Building Large Databases

With Database Manager you can create very large databases. If you do this, keep in mind:

- The size of the database that you can construct is limited by the amount of memory you have available in your computer.
- Plan your database carefully. A database that implements a good scan time and phasing scheme will provide better system performance than the same database that has its blocks scanned all at once.
- iFIX provides a number of features that can aid in processing large databases:
 - Exception-based processing (triggered by a change at the I/O address).
 - One shot processing.

- Long scan times (up to 24 hours with phasing configurations of up to HRS:HRS, HRS:MIN, MIN:MIN, MIN:SEC, and so forth).
- Subsecond processing.

For more information on these topics, refer to the section [Scan Times](#).

Saving Large Databases

Make sure you have enough disk space available when saving your database to disk. If your SCADA server does not have enough disk space, you may lose the changes you have just made. For example, if you build a 3MB database, make sure you have 3MB of free disk space available.

Working with Analog and Digital Blocks

When planning your automation strategy you may want to consider using Analog Input and Digital Input blocks, or Analog Register and Digital Register blocks. All four blocks combine read/write capabilities and eliminate the need for separate output blocks.

Analog Register and Digital Register blocks also have specific addressing and configuration requirements:

- Neither block can cross a poll record boundary.
- You select signal conditioning for an Analog Register block when you configure it. This selection applies to all picture objects associated with the block.
- Offsets to Float or Long data types must be in 2-word (4 byte) increments. This adjusts for the 32-bit word size of these data types.
- Neither block should be used with exception-based poll records.

Although we do not recommend it, if you choose to use Analog Register or Digital Register blocks with exception-based processing, verify that no database block or offset address points to the same address as an exception-based database block. For further information, refer to your I/O driver manual.

If your I/O driver or OPC server supports structured data types (Timers, Counters, etc.), these data types only support an offset of 0 for Analog and Digital registers.

Working with Database Blocks

Your main task when setting up a SCADA server is creating blocks for your process database. In general, you can create a block by completing its dialog box. However, in order to configure a block, you should understand the basic database concepts, such as scan times and phasing. This chapter discusses these and other database concepts.

- [Block Types and Descriptions](#)
- [Block Fields](#)

- [Scan Times](#)
- [Phasing](#)
- [Block Modes](#)
- [Placing Blocks On and Off Scan](#)
- [I/O Drivers](#)
- [Understanding Signal Conditioning](#)

Block Types and Descriptions

Typically, every SCADA server comes with two types of blocks: primary and secondary. The main difference between these block types is that *primary blocks* have scan times and are first in a chain. *Secondary blocks* do not have scan times and are never first in a chain.

You can also purchase the following types of optional blocks:

Database Options

The option...	Provides...	Refer to the section...
Control	Continuous, PID, direct, and digital control.	Understanding Control Blocks to learn more about control blocks.
Statistical Process Control (SPC)	Statistical data analysis and calculations, alarming, supervisory control, and display of statistical data.	Understanding Statistical Process Control Blocks to learn more about SPC blocks.
Batch	State-driven, interlocked, and batch control.	Understanding Batch Blocks to learn more about batch blocks.
SQL	Read and write access to a relational database on a remote server.	Understanding SQL Blocks to learn more about SQL blocks.

The following table summarizes the differences among all the blocks. For detailed information on blocks, refer to the [iFIX Database Reference](#).

Block Summary

Block Type	Primary	Secondary	Standard	Optional
Analog Alarm (AA)	x		x	
Analog Input (AI)	x		x	
Analog Output (AO)	x		x	
Analog Register (AR)	x		x	
Boolean (BL)	x		x	
Calculation (CA)		x	x	
Dead Time (DT)		x		x (Control)
Device Control (DC)	x			x (Batch)
Digital Alarm (DA)	x		x	
Digital Input (DI)	x		x	
Digital Output (DO)	x		x	
Digital Register (DR)	x		x	

Event Action (EV)		x	x	
Extended Trend (ETR)		x	x	
Fanout (FN)		x	x	
Histogram (HS)		x		x (SPC)
Lead Lag (LL)		x		x (Control)
Multistate Digital Input (MDI)	x		x	
On-Off Control (BB)	x			x (Control)
Pareto (PA)	x			x (SPC)
PID (PID)		x		x (Control)
Program (PG)	x			x (Batch)
Ramp (RM)	x			x (Control)
Ratio/Bias (RB)		x		x (Control)
Signal Select (SS)		x		x (Control)
SQL Data (SQD)		x		x (SQL)
SQL Trigger (SQT)	x			x (SQL)
Statistical Control (SC)		x		x (SPC)
Statistical Data (SD)	x			x (SPC)
Text (TX)	x		x	
Timer (TM)		x	x	
Totalizer (TT)		x	x	
Trend (TR)		x	x	

NOTE: The tag count limit per tag type is 32,767.

Understanding Primary Blocks

Primary blocks receive data from an I/O driver or OPC server and generate alarms based upon this information. Primary blocks are usually associated with one or more pieces of process hardware. For example, a pump, a tank, a temperature sensor, a photo cell, a limit switch are all process hardware with which you might associate a primary block.

Most primary blocks, listed in the following table, also include a *scan time*. The scan time controls when SAC scans the blocks in the database. Refer to the section [Scan Times](#) to learn more about them.

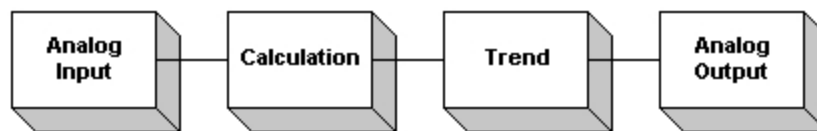
Standard Primary Blocks

The block...	Function:
Analog Alarm (AA)	Provides read/write access to analog data and lets you set and acknowledge alarms.
Analog Input (AI)	Provides read/write access to analog data and lets you set alarm limits.
Analog Output (AO)	Sends analog data to an I/O driver or OPC server when the upstream block, an operator, a Program block, a script, or an Easy Database Access (EDA) program supplies a value.
Analog Register (AR)	Provides read/write access to analog data only when a Data link connected to the block appears on an operator display.
Boolean (BL)	Performs boolean calculations based upon up to eight inputs.
Digital Alarm (DA)	Provides read/write access to digital data and lets you set and acknowledge alarms.

Digital Input (DI)	Provides read/write access to digital data and lets you set alarm limits.
Digital Output (DO)	Sends digital data to an I/O driver or OPC server when the upstream block, an operator, a Program block, a script, or an Easy Database Access (EDA) program supplies a value.
Digital Register (DR)	Provides read/write access to digital data only when a Data link connected to the block appears on an operator display.
Multistate Digital Input (MDI)	Retrieves digital data for up to 3 inputs from an I/O driver or OPC server, combines the inputs into one raw value, and lets you set alarm limits.
Text (TX)	Lets you read and write a device's text information.

Understanding Secondary Blocks

Secondary blocks manipulate data according to your instructions. Secondary blocks usually receive input from an upstream or primary block and perform a specific function with that input, such as perform a calculation or store several successive input values. Therefore, a secondary block cannot be the first block of a chain. However, you can connect secondary blocks to create a chain like the one shown in the following figure.



Chain Showing Secondary Blocks

Note that the first block in the chain is a primary block. This block is the primary source of data for the next block in the chain and determines the scan time for the entire chain. The following table lists the available standard secondary blocks.

Standard Secondary Blocks

The block...	Function:
Calculation (CA)	Performs mathematical calculations using values from the upstream block and up to seven other constants or block values.
Event Action (EV)	Monitors values or alarm conditions of the upstream block and performs actions based on the upstream block's output.
Extended Trend (ETR)	Allows you to collect up to 600 real-time values from an upstream block. You can display these values as a graph by adding charts to your pictures.
Fanout (FN)	Sends the value it receives from its upstream block to up to four additional blocks.
Signal Select (SS)	Samples up to six inputs, manipulating the inputs according to a user-selected mode, and outputs a value to the next block.
Timer (TM)	Counts time by incrementing or decrementing its value.
Totalizer (TT)	Maintains a floating-point total for values passed to it from upstream blocks. This block sends values up to six digits in precision to other blocks. It can display up to fifteen digits of precision in an operator display.
Trend (TR)	Allows you to collect up to 80 real-time values from an upstream block. You can display these values as a graph by adding charts to your pictures.

Understanding Control Blocks

Control blocks provide continuous, direct, or digital control capability. The following table lists the available control blocks.

Control Blocks

The block...	Function:
Dead Time (DT)	Delays the transfer of an input value to the next block in the chain for up to 255 seconds. It can store up to 60 values of incoming variables and sends values on a first in/first out basis.
Lead Lag (LL)	Lets you simulate process dynamics and includes a digital approximation of the exponential equations for lead lag. This block is useful in feed-forward strategies.
PID (PID)	Compares analog inputs to a user-defined set point and sends out incremental adjustments to bring the process variable closer to the set point.
On-Off Control (BB)	Receives analog values and outputs digital values.
Ramp (RM)	Increases or decreases values to a target value at a specified rate. You can enter the target values manually or they can be retrieved from other blocks. You can define three distinct stages for the ramp process.
Ratio/Bias (RB)	Lets you change incoming signals by adding a constant (bias) and/or by multiplying a constant (ratio) after subtracting an offset from the signal. This block uses less memory and executes faster than the Calculation block.

Understanding Statistical Process Control Blocks

Statistical Process Control (SPC) provides statistical data analysis and calculations, alarming, supervisory control, and display of statistical data. The following table lists the available SPC blocks.

Statistical Process Control Blocks

The block...	Function:
Histogram (HS)	Displays an input value's frequency of occurrence.
Pareto (PA)	Accepts, calculates, and sorts the frequency of up to eight input values. You can display these values in an operator display with filled rectangles to create a bar chart.
Statistical Control (SC)	Adjusts a process variable based on calculations of the average offset and the rate of deviation from a target value, XBARBAR. This block is activated if an alarm is generated by a Statistical Data block.
Statistical Data (SD)	Observes data from operator input or other blocks and performs statistical calculations. This block allows for alarming based on standard SPC techniques.

Understanding Batch Blocks

Batch blocks are specifically designed for discontinuous (state-driven, sequenced, interlocked, and batch) control operations. The following table lists the available batch blocks.

Batch Blocks

The block...	Function:
Device Control (DC)	Coordinates the opening and closing of digital devices based upon user-defined conditions.
Program (PG)	Runs short programs for batch operations or to increase the degree of automation in an application.

Understanding SQL Blocks

SQL blocks read and write data to a relational database. The following table lists the available SQL blocks.

SQL Blocks

The block...	Function:
SQL Data (SQD)	Identifies the data to send and retrieve between the process database and your relational database.
SQL Trigger (SQT)	Triggers the execution of SQL commands and defines how your relational database interacts with the process database.

For more information about how these blocks work and how to use them, refer to the [Using SQL](#) manual and the [iFIX Database Reference](#).

Understanding Database Dynamos

iFIX can process information from one or more Database Dynamo objects, also known as loadable blocks. Each Database Dynamo is an optional block that adds functionality to the process database. By using Database Dynamos, you can create new blocks tailored to your needs. For example, you might create a Dynamo that provides a custom PID or other control algorithm.

You can create a Database Dynamo by using the Database Dynamo Toolkit. After you create one, iFIX treats your Database Dynamo like any other block in the process database. This feature enables iFIX to process alarms from the Dynamo, along with the other alarms in the system. Database Dynamos also enable you to:

- Access the Dynamo's fields from any iFIX application.
- Use Database Manager to create, configure, and manage the operation of the Dynamos in the process database.

GE also makes the following Database Dynamos available in the iFIX product:

iFIX Database Dynamos

The Database Dynamo...	Function:
Analog Input with Freeform Scaling (AIS)	Essentially the same as an Analog Input block, but will also calculate a slope for EGU limit values when signal conditioning does not apply.
Analog Register 2 (AR2)	Reads and writes analog values to process hardware. Use the AR2 block if you exceed the limit for the number of AR blocks and need a loadable block with similar functionality.
Breakpoint Lin-	Converts a level measurement from a non-linear tank into a volume measurement.

earization (BPL)	
Transition Counter (CTR)	Counts the transitions of a digital signal.
Digital Register 2 (DR2)	Reads and writes digital values to process hardware. Use the DR2 block if you exceed the limit for the number of DR blocks and need a loadable block with similar functionality.
Extended Trend (ETR)	Allows you to collect up to 600 real-time values from an upstream block. You can display these values as a graph by adding charts to your pictures.
16 Bit Digital Status (D16)	Monitors up to 16 bits in a digital register, and enables or disables 16 independent messages.
Group Alarm (GAB)	Organizes and consolidates the display of alarm information within a large system.
Signal Generator (GEN)	Demonstrates, simulates, or tests. A time-based primary block.
Interval Timer (ITM)	Provides a way to time intervals (normally a digital input). It also includes a GATE input that can be used to disable the timing without having to place the chain off scan.
Momentary Digital Output (MDO)	Sends a brief pulse to a digital output block (for example, to start a motor).
Optimized Momentary Digital Output (ODO)	Sends a brief pulse to a digital output block (for example, to start a motor).
Persistent Array (PAR)	Provides a way to hold up to 60 numeric or text values, and 60 descriptions of those values. A primary block that operates only in manual mode. Values are updated only when you explicitly write to it.
Improved PID (PI2)	Essentially the same functionality as the PID block, with more options.
Time Date Stamp (TDS)	Takes a snapshot of the time of an event.
Text Register (TXR)	Monitors devices where the raw data is an ASCII string. Similar to the Text block (TX), but not scanned.
Text Lookup Block (TXT)	Monitors a status or command register in a PLC. Allows the operator to see and enter meaningful strings.

Database Dynamo Configuration Utility (BTKCFG.exe)

Use the Database Dynamo Configuration Utility (BTKCFG.exe) to add the Database Dynamos in the above table to iFIX. After iFIX is restarted, you will then be able to add blocks of these types in the iFIX Database Manager.

► To add a Database Dynamo:

1. Double-click the Btkcfg.exe file to launch the Database Dynamo Configuration Utility. By default, this file is located in the iFIX install folder, C:\Program Files (x86)\GE\iFIX.
2. In the Available Database Dynamos list, select the Dynamo that you want to add and click Add (or optionally, click Add All). This moves the specified Database Dynamos to the Configured Database Dynamos list.
3. On the File menu, click Save.
4. Restart iFIX. After iFIX restarts, you will then be able to add the blocks in the above table within the iFIX Database Manager.

IMPORTANT: Use caution in removing Database Dynamos (loadable blocks). If you use the Database Dynamo Configuration Utility (BTKCFG.exe) to remove a loadable block but do not remove the block from the iFIX database, an error message appears in the Alarm History that a block type is missing and the blocks are not loaded.

► To remove a Database Dynamo:

1. Double-click the Btkcfg.exe file to launch the Database Dynamo Configuration Utility. By default, this file is located in the iFIX install folder, C:\Program Files (x86)\GE\iFIX.
2. In the Configured Database Dynamos list, select the Dynamo that you want to remove and click Remove (or optionally, click Remove All). This moves the specified Database Dynamos to the Available Database Dynamos list.
3. On the File menu, click Save.
4. Restart iFIX.

► To change a slot number of a Database Dynamo:

1. Double-click the Btkcfg.exe file to launch the Database Dynamo Configuration Utility. By default, this file is located in the iFIX install folder, C:\Program Files (x86)\GE\iFIX.
2. In the Configured Database Dynamos list, select the Dynamo that you want to change the slot number for.
3. In the Slot Number field, enter a new number. You can enter any number between 50 to 149 that is not in use. For more information on slot numbers, refer to the [Understanding Slot Numbers](#) section.
4. On the File menu, click Save.
5. Restart iFIX.

Understanding Slot Numbers

The database contains approximately 100 slots into which [Database Dynamo blocks](#) can be added. These slots are numbered from 50 to 149. Slots 1 to 49 are reserved for GE database blocks.

The Database Dynamo Configuration Utility generates a default slot for the Database Dynamo that you are adding to the database. Use this slot. If you must change the Database Dynamo slot, select the Dynamo in the configured column and enter a new value for the slot in the Slot field.

NOTE: Once a slot is selected, it should never be changed. If you must change the slot, export the database to an ASCII file first. Change the slot of the block. Delete the database and import the ASCII file into a new database.

Block Fields

Whenever you add a block, its configuration dialog box appears. The dialog box fields and controls represent locations in the block called *fields*. These fields store the information you enter into the dialog box. This information includes such data as the block's name, description, scan time, I/O address, and scan status.

Other block fields receive information from your process hardware or from other blocks. For example, a primary block's current value comes from an I/O device. However, a secondary block receives its current value from an upstream block.

All block fields use a common naming convention:

format _ name

The format indicates the type of data that the field stores. The following table lists the available formats:

Field Formats

Format:	Description:	Used in...
A_	ASCII Format.	Data links and objects in pictures.
F_	Floating-point Format.	Data links, objects in pictures, and block-to-block references.
E_	15-Digit Precision Format.	Data links, objects in pictures, and block-to-block references. Valid values range from +/- 3.40282300000000e+/-38, with 15 digits of accuracy.
T_	Graphic Format.	Charts.

Refer to the [Creating Pictures](#) manual for more information on Data links, objects in pictures, and charts.

The name indicates the specific information in the field. For example, the current value of a block is identified by the name:

CV

The combination of the field format and the field name provides you with the information you want. For example, if you want the current value of a block displayed as a number, you select the field F_CV. If you want the current value of a block displayed as text, you select the field A_CV.

Locating Block Fields

You can display a list of fields for the currently selected block using the Expression Builder or the [iFIX Database Reference](#) help. Using this information, you can configure Data links or other objects in your operator displays to extract block field data by specifying a data source. Alternatively, you can write a Visual Basic for Applications (VBA) script or an Easy Database Access program to extract and display block field data. For more information about the Expression Builder, specifying data sources, and adding Data links and objects, refer to the [Creating Pictures](#) manual.

Completing Block Dialog Boxes

The [iFIX Database Reference](#) provides information to help you complete block dialog boxes. The remainder of this chapter provides related information for completing common block fields, such as the scan time, phase, and I/O driver fields. Refer to these sections to learn how to configure these fields for your blocks.

Scan Times

All primary blocks have a *scan time*. The scan time determines how often SAC processes the block and sends the current value to the next block in the chain. SAC processes all secondary blocks chained to a primary block according to the primary block's scan time.

SAC can process a chain using one of the following methods:

- [Time-based processing](#)
- [Exception-based processing](#)
- [One shot processing](#)

Time-based processing is best used when you want to regularly scan a block. If you only need to scan a block when its value changes, use exception-based processing instead. Similarly, you can use one shot processing if you need to scan a block when the process database initially loads.

Understanding Time-Based Processing

In time-based processing, SAC processes a block at a set time. The following table lists the scan time ranges you can enter for time-based chains.

Scan Time Ranges	
Range	Increments by...
5 to 95 subseconds (0.05 to .95 seconds)	.05 seconds (0.05, 0.10, 0.15, 0.20, and so forth.)
1 to 60 seconds	1 second
1 to 60 minutes	1 minute
1 to 24 hours	1 hour

SAC scans chains with hour and minute scan times based on the system clock of the local SCADA server. Scan times are set relative to midnight (00:00:00 hours). SAC scans chains with second and subsecond scan times based on the computer's start up time, as the following tables describes.

Time-Based Scan Time Examples	
When a block has a scan time of...	SAC processes it every...
1 hour	Hour on the hour.
1 minute	Minute on the minute.
10 seconds	10 seconds from your computer's start-up time.

Assigning Time-Based Scan Times

You can assign a time-based scan time to a block by completing its Scan Time field with the following format:

`time unit`

The following table lists the valid units and their abbreviations. If you do not enter a unit of time, iFIX assumes the unit is seconds.

Unit	Entry
Minutes	M
Hours	H

Example: Assigning Time-Based Scan Times

To scan a block every 3 hours, enter:

`3H`

Because SAC processes this scan time based to the system clock, it scans the block at 0:00, 3:00, 6:00, 9:00, and 12:00 regardless of when you place it on scan.

Follow these guidelines when you assign scan times:

- Assign scan times larger than the poll rate assigned in the I/O driver. This ensures that the I/O driver has time to read and send new values to SAC before SAC scans each block again. See your I/O driver manual for more information about the poll rate.
- Phase (stagger) scan times to reduce the risk of overloading the CPU. Refer to the section [Phasing](#) to learn more about phasing.
- Assign critical process chains a more frequent scan time than non-critical chains. If you need to have a chain scanned every 2 minutes, assign a 2-minute scan time, not a 5-second scan time. Remember that very short scan times require more CPU time and SAC processing than longer scan times.
- If a chain does not need processing at a set time, assign exception-based processing. Doing so will require less CPU time and improve performance.

Understanding Exception-Based Processing

Exception-based processing lets SAC scan a block or chain by exception, not at scheduled time intervals. An *exception* is:

- A change in a process value greater than the defined exception dead band; or
- An unsolicited message from your process hardware.

Using exception-based processing generally requires less CPU time and improves performance because SAC does not have to scan blocks at defined intervals. However, if a block's value changes very frequently, time-based processing may be more efficient.

You can use exception-based processing only if your I/O driver supports it. Consult your I/O driver client manual to learn if your I/O driver supports exception-based processing.

NOTE: The SM2 driver supports exception-based processing; the SIM driver does not support exception-based processing.

CAUTION: Do not assign the same I/O address to exception-based and time-based blocks. Doing so will cause the exception-based blocks to occasionally miss an incoming value.

Blocks that Work Best with Exception-Based Processing

While you can use exception-based processing with most blocks, certain blocks perform better with exception-processing than others. You can use the following blocks in exception-based chains as needed:

• Analog Input	• On-Off Control
• Analog Output	• Pareto
• Digital Input	• Ratio/Bias
• Digital Output	• SQL Data

- Fanout
- Histogram
- Timer
- Totalizer

Using Program and Statistical Data Blocks

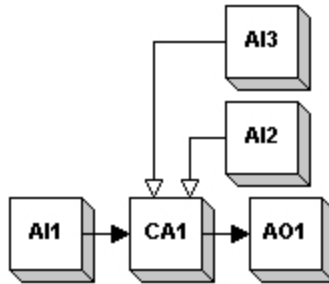
Typically, the blocks listed in the following table are used as stand-alone blocks and have limitations when incorporated in an exception-based chain.

Program and Statistical Data Block Considerations	
When you add the block...	In an exception-based chain, do not use the...
Program	CALL, DELAY, WAITFOR, or WAITSTAT commands.
Statistical Data	WAIT TIME fields.

Blocks that Receive Multiple Inputs

The Boolean, Calculation, Event Action, and Signal Select blocks can use values from multiple blocks, but SAC only processes them according to their upstream primary block's scan time. Therefore, use these blocks with care in exception-based chains.

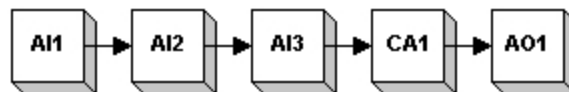
For example, the following figure shows a sample chain containing a Calculation block that receives values from Analog Input blocks outside the chain.



Exception-Based Chain Processing

In this chain, if SAC scans AI1 by exception, then CA1 only accesses values from the remaining Analog Input blocks when that exception occurs. CA1 does not access values based upon the scan times of the Analog Input blocks outside the chain. This means that regardless of whether AI2 and AI3 change in value, CA1 does not recalculate its output unless triggered by AI1.

The following figure shows an improved chain design using the blocks from the previous figure. This chain allows CA1 to recalculate its value whenever an exception occurs to any of the Analog Input blocks and ensures all blocks are processed before the recalculating the output.



Improved Exception-Based Chain Processing

Blocks that Define a Time Constant

PID, Lead Lag, and Dead Time blocks use the local computer's system time to define a time constant. For this reason, it is recommended not using them in exception-based chains.

Using Analog Alarm and Digital Alarm Blocks

Analog Alarm and Digital Alarm blocks support exception-based processing. However, you must leave the Re-alarm Time and the Delay Time fields unmodified; otherwise, SAC places these blocks (and their chains) off scan when iFIX starts or when you reload the database. SAC processes exception-based Analog Alarm and Digital Alarm blocks only when an operator

acknowledges an alarm from the iFIX WorkSpace. If an operator acknowledges the alarm with the Remote Acknowledge field, SAC does not process the block.

Assigning Exception-Based Scan Times

You can assign an exception-based scan time to a block by selecting Process by Exception from the block's dialog box. If you are configuring an exception-based chain with multiple primary blocks, you must enter 0 in the Scan Time field of every primary block that does not start the chain, and select the Off Scan option button.

Configuring the primary blocks that do not start the chain in this manner ensures that SAC processes the chain properly.

Understanding One Shot Processing

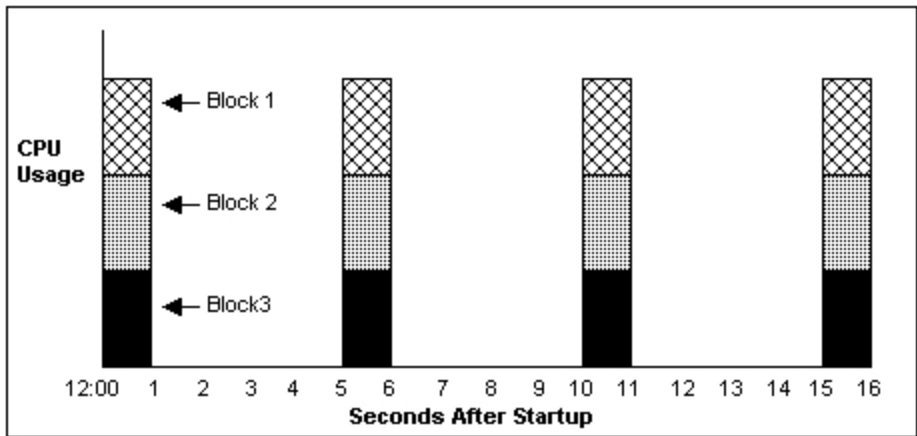
You can configure any primary block or chain to be scanned once on start-up by setting the scan time to zero and placing the block on scan. Once scanned, SAC does not scan the block or chain again until it restarts, the database is reloaded, or you place the block off scan and then on scan.

Any block with a scan time may be configured for one shot processing. However, the same restrictions that apply to exception-based chains also apply to one shot-based chains.

Phasing

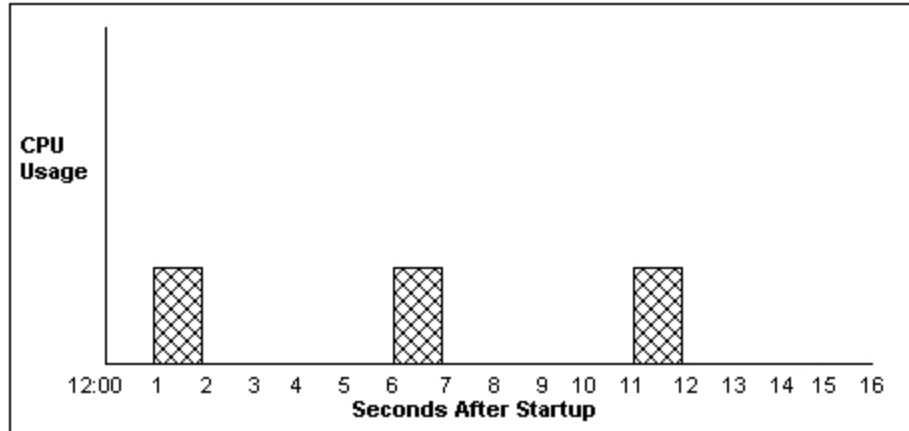
Phasing staggers the time at which SAC scans your blocks. This is particularly important for large databases because it can dramatically improve performance. Even for small databases, phasing blocks results in more efficient use of CPU time.

For example, if you have 3 unphased blocks with a 5-second scan time, SAC processes all 3 blocks at the same time. The following figure shows the CPU usage when SAC processes these blocks simultaneously.



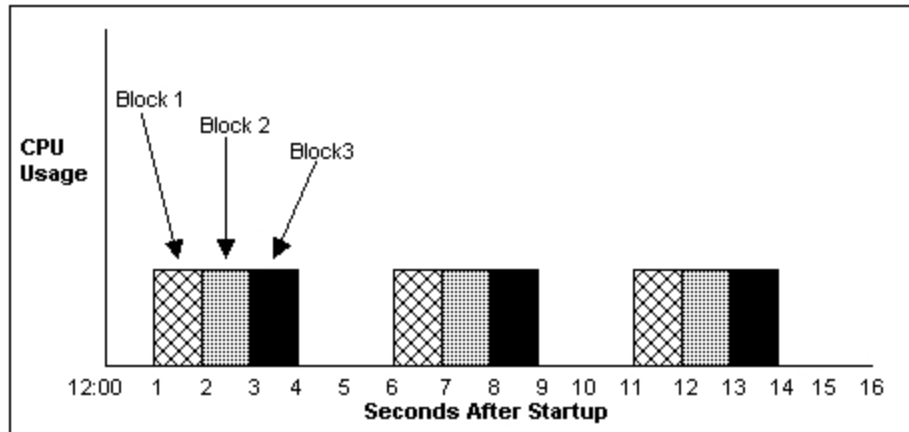
Processing Blocks Without Phasing

By phasing these blocks, you specify when SAC processes each block. For example, if you specify a 1-second phase for the first block, SAC scans the block as the following figure shows:



Phasing for Block 1

If you specify a 2-second phase for Block 2, and a 3-second phase for Block 3, you can avoid processing the blocks simultaneously and evenly distribute the block processing and the CPU work load, as the following figure shows.



Phasing Block Scan Times for Multiple Blocks

Phasing Second and Subsecond Blocks

Normally, when you specify a phase for a block with a second or subsecond scan time, SAC offsets the block's initial scan by the phase time when iFIX starts or when the database is reloaded. However, SAC ignores the phase when:

- A Program or an Event Action block places the phased block on scan.
- An operator or script places the block on scan.
- An EDA program places the block on scan.

Assigning Phase Times

You can define the phase time by entering it into the Phase field. The phase you enter must have the same unit or one unit lower than the scan time unit, as the following table describes.

Phase Time Formats	
If the Scan Time is in...	The Phase Time must be in...
Hours	Hours:Minutes
Minutes	Minutes:Seconds
Seconds	Seconds
Subseconds	Subseconds

For example, if the scan time is 5 minutes, you enter a phase of 1 minute and 30 seconds as follows:

1:30

To specify a 30-second phase instead, enter 0:30.

Overphasing

You can also *overphase* your blocks. An overphased block is one that has a larger phase than scan time. For example, assume you have the chain shown in the [Exception-Based Chain Processing](#) figure. In this chain, AI2 and AI3 have 5-second scan times. AI1 is overphased with a 5-second scan time and a 10-second phase. This delays AI1 by 10 seconds and ensures that SAC scans AI2 and AI3 before AI1. This means that the Calculation block receives the most recent values on which to perform its functions.

Overphasing and Long Scan Times

SAC only lets you overphase blocks with second or subsecond scan times. For blocks with a scan time of a minute or more, you can create an initial offset from the time that SAC would normally scan the block. For example, assume you want to scan a block every 6 hours with an offset of 2 hours and 10 minutes. You can configure the offset by entering the following phase in the Phase field:

2:10

Because the scan time is linked to the system clock, SAC always scans the block at 2:10, 8:10, 14:10, and 20:10 regardless of when it was placed on scan. Also remember, SAC scans blocks with scan times of one minute or longer immediately when iFIX starts or when the database is reloaded. To forego this initial scan, start SAC with the S command line parameter. Refer to the section [Controlling SAC Startup](#) in the Setting up the Environment manual for more information on specifying SAC command line parameters.

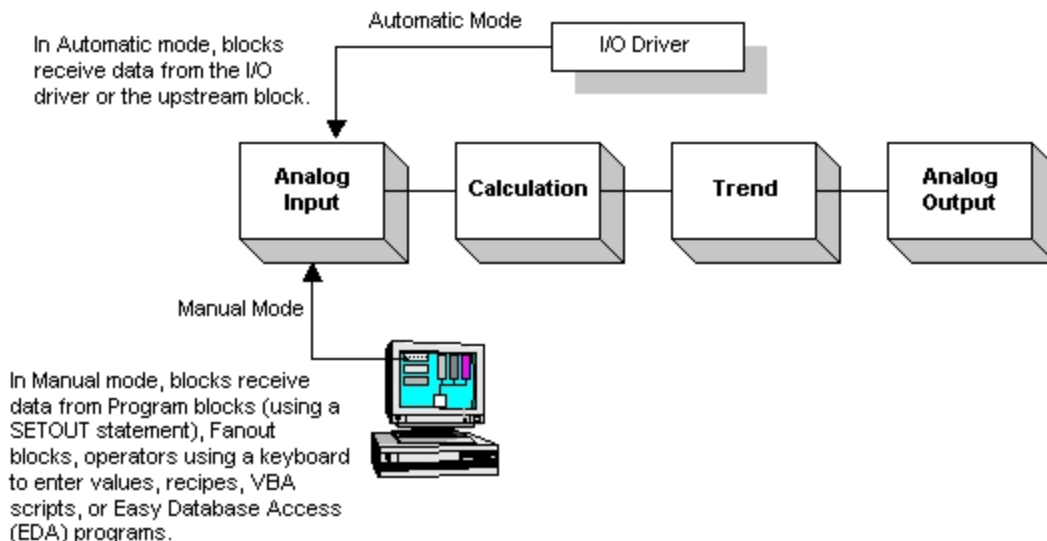
Block Modes

Blocks can receive data from the following sources:

- Other blocks.
- An I/O driver or OPC server.
- The keyboard (using Data links).
- Recipes.

- Easy Database Access programs.
- Visual Basic for Applications (VBA) scripts.

You can control from where a block receives data by setting its *mode*, as the following figure shows:



Placing a block in Manual mode does not place it off scan; SAC still processes the block using its last value. Manual mode only prevents a block from receiving data from an I/O driver, an OPC server, or the upstream block. SAC also scans exception-based blocks when they change mode (from Automatic to Manual mode or vice versa). While in Manual mode, SAC accepts any input within the EGU range and immediately scans the block. No deadband checking is made on these values.

You can toggle a block's mode from an operator display by adding an object to the display and then using the Toggle Tag Auto/Manual Expert from the Task Wizard. To display a block's mode from an operator display, add a Data link that references the block's A_AUTO field. If you have enabled in-place data entry when configuring the data link, you can then modify that block's mode from the operator display by clicking on the Data link and entering AUTO or MANL. You can also modify a block's mode from the Database Manager by entering AUTO or MANL in the Curr Mode column for the selected block.

NOTE: Blocks perform alarming in both Automatic and Manual modes.

The exact function of certain blocks depends on their mode, as the following table describes

Blocks in Automatic and Manual Modes

The block...	In automatic mode...	And in manual mode...
Analog Input	Reads and writes data every scan cycle.	Receives data from operators entering values, scripts, Program blocks, recipes, or Easy Database Access (EDA) programs, but does not output values.
Device Control	Executes all statements without interruption.	Suspends execution until the block is placed in Automatic mode.
Digital Input	Reads and writes data every scan cycle.	Receives data from operators entering values, scripts, Program blocks, recipes, or EDA programs, but does not output values.
PID	Performs PID adjustments. Set points can be changed through a Data link in this mode.	Suspends automatic PID outputs. This allows you to change the output from the block. Block values, such as the set point and tuning parameters, can be changed from the keyboard.

Program	Executes all statements without interruption.	Suspends execution of its statements until the block is placed in Automatic mode. When the block is switched back to Automatic mode, it continues from where it stopped.
Statistical Control	Feeds information back into your process when the Statistical Data block generates an alarm.	Suspends feedback.
Statistical Data	Performs online statistical process control on incoming data from other blocks.	Suspends online data observations. Used for off-line statistical process control.

Understanding PMAN Mode

As blocks change from Automatic and Manual modes, they enter a pending state. This state indicates that SAC has acknowledged the mode switch and will change it on the next scan.

Blocks placed into Manual mode enter PMAN mode. The block remains in this state until SAC scans the block again. While in PMAN mode, SAC continues to scan the block according to its scan time and phase. Manual entries from an operator or a Program block force SAC to scan the block immediately and put it into Manual mode. Scans after a manual entry depend on the block's scan time.

Example: Blocks in PMAN Mode

For blocks with short scan times, the next scan time occurs relative to the manual entry. For example, suppose SAC scans the block AI1 at 1:15:30. This block has a 30-second scan time and a 5-second phase. If an operator puts AI1 into Manual mode and enters a value at 1:15:45, SAC scans the block immediately and resets AI1's next scan to 1:16:15 and every 30 seconds thereafter. If the operator subsequently enters another value at 1:15:50, SAC scans AI1 again and reset AI1's next scan to 1:16:20 and every 30 seconds thereafter.

For blocks with scan times of a minute or longer, SAC synchronizes the next scan to the system clock. For example, assume AI2 has a one hour scan time and a 30-minute phase. If an operator placed the block into Manual mode and enters a value at 15 minutes past the hour, SAC scans AI2 immediately and on the half hour.

Understanding Blocks in PAUT Mode

Blocks placed in Automatic mode enter PAUT mode. The block remains in this state until SAC scans the block again. While in PAUT mode, SAC scans the block as if it were in Automatic mode.

Example: Blocks in PAUT Mode

For blocks with short scan times, this means that the next scan time occurs relative to the last scan. For example, suppose SAC scans AI1 at 1:15:30. This block has a 30-second scan time. If an operator changes the block's mode to Automatic, SAC:

1. Puts the block into PAUT mode.
2. Scans the block again at 1:16:00, placing it in Automatic mode.

For blocks with long scan times, SAC synchronizes the next scan to the system clock. For example, assume AI2 has a one hour scan time and a 30-minute phase. If an operator places the block into Automatic mode, SAC:

1. Puts AI2 into PAUT mode.
2. Scans the block again on the half hour, placing it into Automatic mode.

Placing Blocks On and Off Scan

SAC processes all database blocks placed on scan when it starts or a database is loaded. By default, Database Manager prompts you to place on scan each block you add. SAC stops processing a block or a chain when:

- From the iFIX WorkSpace, you select an object connected to a primary block and you run a script that places the block off scan.
- You modify a block with Database Manager. Modifying a block while SAC is processing that block's chain places the entire chain off scan. As an option, you can set up Database Manager to automatically place the modified block and its chain on scan when you finish editing it. Refer to the section [Setting Database Manager Preferences](#) to learn more about Database Manager options.
- You delete any block within a chain. Refer to the section [Deleting Blocks](#) to learn more about removing blocks from the process database.
- Statements in Program blocks, Event Action blocks, or an EDA program place a chain's primary block off scan.
- You connect the blocks in a chain incorrectly and SAC cannot scan the chain. Use Database Manager's Verify command to determine which blocks are improperly chained. Refer to the section [Verifying Databases](#) to learn about examining your database for errors.
- A Program block finishes processing its statements and exits, the Program block does not execute again until the database is reloaded or the block is turned off and placed back on scan.
- You can place a block off scan from the Scan Status column displayed in the spreadsheet.
- An operator clicks an object with a VBA script that places a block off scan. You can quickly add such a script to an object with the Turn Tag Off Scan button. To use this button, you must first drag it from the CommandTasks toolbar category onto a toolbar. To learn how to display toolbar categories and add a button to a toolbar, refer to the section [Customizing Toolbars](#) in the Understanding iFIX manual.

Placing a primary block off scan turns that block's entire chain off scan. You can put the chain back on scan by:

- Opening an operator display, selecting an object connected to the chain's primary block, and running a script that places the block on scan.
- Setting up a Program or an Event Action block that places the chain back on scan.
- Changing the text in the Scan status column from OFF to ON.
- Adding or modifying a block with Database Manager or the iFIX WorkSpace. Depending on the options selected, both programs can automatically place the block on scan.
- An operator clicks an object with a VBA script that places a block on scan. You can quickly add such a script to an object with the Turn Tag On Scan button. To use this button, you must first drag it from the CommandTasks toolbar category

onto a toolbar. To learn how to display toolbar categories and add a button to a toolbar, refer to the section [Customizing Toolbars](#) in the Understanding iFIX manual.

Blocks with Long Scan Times

Blocks that have long scan times react differently to on/off scan changes than blocks that have short scan times. If you change the block's scan status, it enters a pending state: PON (pending on) or POFF (pending off). This state indicates that a scan status change was requested and is pending, but SAC has not placed the block off or on scan. While a block is in PON, new values are ignored.

Example: Placing Blocks with Long Scan Times On Scan

Assume you have a block with a one hour scan time and you place it on scan 45 minutes into its scan cycle. The block enters the PON state and remains there for 15 minutes in order to synchronize it with SAC. Once synchronized, SAC changes the block's state and places it on scan.

Using the S Parameter

By default, SAC changes a block's scan status shortly after entering the pending state. However, if SAC was started with the "S" command line parameter, the block remains in the pending state until SAC is ready to scan it.

I/O Drivers

In order for each primary block in the database to receive data, you must connect to your I/O using an I/O driver. The driver you select depends on your process hardware. GE sells drivers for many types of hardware. Contact your GE Sales Representative, or refer to our web site at <https://digitalsupport.ge.com> for a list of available drivers.

After you purchase a driver and install it, you can start specifying I/O points you want the current block to use. If the I/O point does not exist, Database Manager starts your I/O driver configuration program so you can add it. Refer to your I/O driver documentation to learn how to add an I/O point to your driver configuration.

iFIX supplies an OPC Client I/O driver, as well as two simulation drivers.

OPC Client Driver

The OPC Client driver provides the interface and communications protocol between OLE for Process Control servers and iFIX.

The OPC Client driver supports the following features:

- Analog register and digital register database blocks
- Special addressing for analog output and digital output blocks
- Text blocks
- Item property I/O addresses for text blocks
- Block writes
- Data arrays
- Exception-based processing
- Latched data

Simulation Drivers

You can use the SIM and SM2 to test your chains before you connect to real I/O. The simulation drivers are matrixes of addresses. Database blocks read values from and write values to these addresses. If one block writes to a specific address, other blocks can read the same value from the same address. You can save these values when you save the process database; however, iFIX removes them from memory when SAC starts or you reload the database.

Both drivers have the following in common:

- Provide a matrix of addresses that database blocks can read from and write to.
- Support analog and digital database blocks.
- Support text blocks.

The drivers differ in the following ways:

The SM2 driver...	The SIM driver...
Provides three independent sets of registers. Analog blocks automatically access the analog registers, digital blocks automatically use the digital registers, and Text blocks automatically access the text registers.	Provides one set of registers shared by both analog, digital, and text blocks.
Changing a register in one set does not change the same register in the other set. For example, if you change the value of the analog register 1000, the value of the digital register 1000 is unchanged.	Changing an analog register in the SIM driver modifies the register for analog, digital, and text reads. For example, if you change the value of the analog register 1000, you also modify the value of the same digital register.
Provides 20,000 analog, 20,000 16-bit digital registers, and 20,000 text registers.	Provides 2000 analog and digital registers, a total of 32,000 bits.
Stores analog values in 4-byte (32-bit) floating point registers, numbered 0 to 19999. Incoming values are not scaled.	Stores analog values in 16-bit integer registers, numbered 0 to 2000. Incoming 32-bit values are scaled to 16-bit values (0 - 65535).
Digital values are stored in 16-bit integer registers, numbered 0 to 19999.	Digital values are stored in 16-bit integer registers, numbered 0 to 2000.
Text values are stored in 8-bit registers numbered 0 to 19999. Each register holds one text character for a total of 20,000 bytes of text.	Text values are stored in the same area as analog and digital values, numbered 0 to 2000.
Provides a register to simulate communication errors.	Cannot simulate communication errors. However, the SIM driver does provide registers RA through RK and RX through RZ to generate random numbers. For more information, refer to the Using Signal Generation Registers in the SIM Driver section.
Supplies a C API that allows applications to access SM2 analog, digital, and text values.	Does not support a C API for accessing SIM values.
Supports exception-based processing.	Does not support exception-based processing.
Supports latched data for Analog Input, Analog Alarm, Digital Input, Digital Alarm and Text blocks when a simulated communication error is enabled.	Does not support latched data.
Can read and write the individual alarm status of each SM2 register.	Cannot read and write the individual alarm status of any SIM register.

Does not provide alarm counters.

Provides alarm counters that show the general alarm state of a SCADA server. For more information, refer to the [Using Alarm Counters](#) chapter of the Implementing Alarms and Messages manual.

Using the OPC Client I/O Driver

The OPC Client I/O driver allows you to bring OPC data into and out of iFIX. It is automatically installed with iFIX; however, if you choose to create a customized install program, the OPC Client I/O driver is an optional component.

► To use the OPC I/O Driver:

1. In the Database Manager, in the Driver field for the primary block, select OPC - OPC Client vx.xx from the list.
2. Click the Browse button next to the I/O Address field. The Browse I/O Address dialog box appears.
3. Select an I/O point from an available server and group, and then click OK to exit the dialog box. The I/O Address field will display the following:

```
Server;Group;ItemID[;AccessPath]
```

When you click out of the I/O Address field, *;No Access Path* will be appended to the existing text, like this:

```
Server;Group;ItemID;NoAccessPath
```

4. If applicable, from the Signal Conditioning list, select a format for mapping the values coming from your process hardware.
5. If applicable, from the Hardware Options list, select a device control addressing format for the database block. This selection is overridden if you select a datatype from the Requested Datatype list on the Item Configuration page of the OPC Client I/O driver Power Tool.

For detailed information, refer to the [OPC Client Driver](#) online help.

Using the Simulation Driver

You may prefer to use the SIM driver over the SM2 driver for one or more of the following conditions:

- Generating a repeating pattern of random and predefined values to help you test your chains.
- Using alarm counters to show the general alarm state of your SCADA server.

► To use the SIM driver:

1. In the primary block's Driver field, type SIM.
2. Complete the I/O Address field with the following syntax:

```
register:bit
```

For analog values, the register ranges from 0 to 1999. The bit is not used.

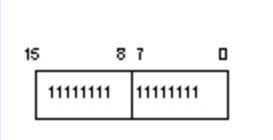
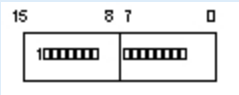
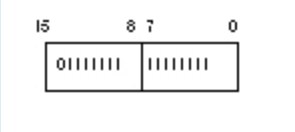
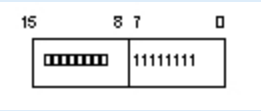
For digital values, the register ranges from 0 to 1999. The bit is 0 to 15. The full range of register/bit settings is 0:0 to 1999:15.

NOTE: The SIM driver does not support analog scaling (A_SCALE_* and F_SCALE_* database fields).

Examples: SIM Addresses

0:0
50:2
63:15

The SIM driver shares only one set of registers for Analog and Digital blocks. As a result, you can address all 2000 registers as analog or digital values. The following table shows the digital bit values when a SIM register contains an analog value.

SIM Analog and Digital Values	
When the Analog Value is...	The Bits for Digital Values are...
65535	Bits 15 to 0 are set to 1 
32768	Bit 15 is set to 1 Bits 14 to 0 are set to 0 
32767	Bit 15 is set to 0 Bits 14 to 0 are set to 1 
255	Bits 15 to 8 are set to 0 Bits 7 to 0 are set to 1 

Database Manager does not accept entries into the Hardware Options and Signal Conditioning fields when using the SIM driver. In addition, the SIM driver supports:

- Only five-digit precision instead of the standard seven-digit precision.
- Time-based processing; you cannot use exception-based processing.

- The output of good values. The SIM driver does not output bad values. If you are testing your system for fault tolerance, remember that the SIM driver does not send communication errors (BAD values).

Using Signal Generation Registers in the SIM Driver

To help you test your database with simulated input, the SIM driver provides a set of registers that generate a repeating pattern of random and predefined values. For example, you could ramp a value to simulate the performance of specific chains or you might generate a series of random numbers to test the entire database.

► To assign one of these registers to a block:

1. Enter **SIM** in the block's Driver field.
2. Complete the I/O Address field with the following syntax by entering a two-letter register acronym as listed in the [SIM Signal Generation Registers](#) table:

```
register:bit
```

The bit portion is needed only when using a digital block.

Example

To ramp a value with the RA register, enter the following text in the I/O Address field:

```
RA
```

The following table lists the available registers.

SIM Signal Generation Registers

The register...	Lets you...	Valid Entry
RA	Ramp a value from 0 to 100% of the EGU range at a rate controlled by the RY register.	Read only
RB	Count from 0 to 65535 at a rate of twenty counts per second.	Read only
RC	Shift one bit through a 16-bit word at a rate controlled by the RZ register.	Read only
RD	Generate a sine wave from 0 to 100% of the EGU range at a rate controlled by the RY register.	Read only
RE	Generate a sine wave from 0 to 100% of the EGU range at a rate controlled by the RY register. The sine wave is delayed 90 degrees relative to the RD register.	Read only
RF	Generate a sine wave from 0 to 100% of the EGU range at a rate controlled by the RY register. The sine wave is delayed 180 degrees relative to the RD register.	Read only
RG	Generate random values between 25% and 75% of the EGU range.	Read only
RH	Ramp up a value to 100% of the EGU range and then ramp it down to 0% again at a rate controlled by the RJ register.	Read only
RI	Control the ramp direction of the value in the RH register. When zero, register RH ramps down; when one, RH ramps up. The value automatically changes when RH reaches 0 or 100% of its EGU value.	Numeric Value (0 or 1)
RJ	Control the ramp speed (in cycles per hour) for the value in register RH. The default value is 60 (1 cycle per minute).	Numeric Value (2 to 3600)
RK	Enable or disable the generation of the value in the RH register. Enter zero to freeze (disable) ramp	Numeric

	and a non-zero value to enable it.	Value (0 or 1)
RX	Enable or disable the generation of values in the other registers. Enter zero to freeze (disable) all registers and a non-zero value to enable all registers.	Numeric Value (0 or 1)
RY	Control the speed (in cycles per hour) at which new values are generated for registers RA, RD, RE, and RF. By default, the RY register is set to 60 (1 cycle per minute).	Numeric Value (2 to 3600)
RZ	Control the speed (in bits per minute) that the register RC changes its value. By default, the RZ register is set to 180 (3 bit shifts per second).	Numeric Value (2 to 1200)

All SIM registers support Analog Input, Analog Register, Digital Input, and Digital Register blocks. However, as the following table describes, certain blocks provide optimum performance when used with certain registers.

The block...	Works best with the register...
Analog Input	RA, RD, RE, RF, RG, and RH
Analog Output	RJ, RY, and RZ
Analog Register	RA, RD, RE, RF, RF, RH, RI, RJ, RK, RX, RY, and RZ
Digital Input	RB and RC
Digital Register	RB, RC, RI, RK, and RX

NOTE: The RB and RC registers support Digital Register offsets of A_0 to A_15.

Using the Simulation 2 Driver

You may prefer to use the SM2 driver over the SIM driver when one or more of the following conditions occur:

- You have more test data than the SIM driver can hold.
- You want to determine how the database responds to 32-bit values.
- You need to access the driver from a C program.

Accessing SM2 Registers

The SM2 driver matrix consists three independent sets of registers, one for analog values, one for digital values, and one for text values. Analog database blocks read from and write to analog registers only. Once a block writes a value, other analog blocks can read the value from the register written to. Digital database blocks work the same way, reading and writing from the digital registers. iFIX clears all SM2 values when iFIX starts.

The SM2 driver does not use the Hardware Options or Signal Conditioning fields.

► **To use the SM2 register:**

1. Enter SM2 in the primary block's Device field.
2. Complete the I/O Address field with the following syntax:

For Analog values: *register*

For Digital values: *register:bit*

For Text values: *register*

SM2 Address Examples		
Analog Examples	Digital Examples	Text Examples
1000	5000:10	2000
16435	23:15	off

Generating Bad Data with the SM2 Driver

The SM2 driver provides an S register to simulate a communication error. Using this register, all analog and digital reads return an error as if communication to the process hardware has been lost.

To use this feature, set the S register to 1.

NOTE: The SM2 driver latches data when a simulated communication error is enabled.

Using the SM2 C API

You can access SM2 analog, digital, and text values through the C API that the driver supplies. The file SM2API.H describes the API and the functions reside in the file SM2API.LIB. You can link this library file to your C application to access the API's functions. You can find both files in your Base path. By default, this path is C:\Program Files (x86)\GE\iFIX\ or C:\iFIX\, depending where you installed iFIX.

NOTE: You must have the iFIX Integration (EDA) Toolkit installed to use this API.

Example

Suppose you are using the SM2 driver to store data from a legacy system. Using the C API and a number of preconfigured analog blocks, you can extract your data from the legacy system and store it in your process database.

C API Functions

Syntax	Values read, written, and returned
UINT16 GetAnalog(UINT16 index, FLOAT *data);	<p>GetAnalog reads an analog value (32-bit float) to the register indicated by 'index'.</p> <p>FE_OK is returned if the operation succeeds.</p> <p>FE_IO_ADDR is returned if the register index is out of range.</p> <p>FE_RANGE is returned if the analog value exceeds the range of a 32-bit float.</p> <p>NOTE: GetAnalog and GetDouble access the same table in the SM2.</p>
UINT16 SetAnalog(UINT16 index, FLOAT data);	<p>SetAnalog writes an analog value (32-bit float) to the register indicated by 'index' and causes an exception for the specified register even if the data has not changed.</p> <p>FE_OK is returned if the operation succeeds.</p> <p>FE_IO_ADDR is returned if the register index is out of range.</p> <p>NOTE: SetAnalog and SetDouble access the same table in the SM2.</p>

UINT16 GetDouble (UINT16 index, DOUBLE *data);	<p>GetDouble reads an analog value (64-bit float) to the register indicated by 'index.'</p> <p>FE_OK is returned if the operation succeeds.</p> <p>FE_IO_ADDR is returned if the register index is out of range.</p> <p>NOTE: GetAnalog and GetDouble access the same table in the SM2.</p>
UINT16 SetDouble (UINT16 index, DOUBLE data);	<p>SetDouble writes an analog value (64-bit float) to the register indicated by 'index' and causes an exception for the specified register even if the data has not changed.</p> <p>FE_OK is returned if the operation succeeds.</p> <p>FE_IO_ADDR is returned if the register index is out of range.</p> <p>NOTE: SetAnalog and SetDouble access the same table in the SM2.</p>
UINT16 GetDi- gital(UINT16 index, UINT16 *data);	<p>GetDigital reads 16 digital values (all 16 bits in one of the 20,000 digital registers) to the register indicated by 'index'.</p> <p>FE_OK is returned if the operation succeeds.</p> <p>FE_IO_ADDR is returned if the register index is out of range.</p> <p>NOTE: The API can only read and write the entire 16 bit digital register at one time. If you want to change 1 bit, you can read the register, modify the desired bit and write the register. However, when you modify a single bit, ensure that only one thread in one application is accessing a digital register at one time.</p>
UINT16 SetDi- gital(UINT16 index, UINT16 data);	<p>SetDigital writes 16 digital values (all 16 bits in one of the 20,000 digital registers) to the register indicated by 'index' and causes an exception for all 16 bits of the specified register even if the data has not changed.</p> <p>FE_OK is returned if the operation succeeds.</p> <p>FE_IO_ADDR is returned if the register index is out of range.</p> <p>NOTE: The API can only read and write the entire 16 bit digital register at one time. If you want to change 1 bit, you can read the register, modify the desired bit and write the register. However, when you modify a single bit, ensure that only one thread in one application is accessing a digital register at one time.</p>
UINT16 SetDi- gitalEx(UINT index, UINT16 data, UINT16 mask)	<p>SetDigitalEx writes 16 digital values (all 16 bits in one of the 20,000 digital registers) to the register indicated by 'index' and causes an exception for specific bits selected from a mask. An exception is triggered for the bits set in the mask even if the data has not changed.</p> <p>FE_OK is returned if the operation succeeds.</p> <p>FE_IO_ADDR is returned if the register index is out of range.</p> <p>NOTE: The API can only read and write the entire 16 bit digital register at one time. If you want to change 1 bit, you can read the register, modify the desired bit and write the register. However, when you modify a single bit, ensure that only one thread in one application is accessing a digital register at one time.</p>
UINT16 GetText (UINT16 index, char *data, int size)	<p>GetText reads the text specified by 'data' from text registers starting at the register indicated by 'index'. The number of characters to read is indicated by 'size'. GetText does not automatically add a null terminator to the text being read. If you require null-terminated strings, make sure your program adds a null terminator after reading text.</p> <p>FE_OK is returned if the operation succeeds.</p> <p>FE_IO_ADDR is returned if the register index is out of range.</p>

UINT16 SetText (UINT16 index, char *data, int size)	<p>SetText writes the text specified by `data` to text registers starting at the register indicated by `index`. The number of characters to write is indicated by `size`. Exception-based processing is not supported for text values. SetText does not automatically add a null terminator to the text being written. If you require null-terminated strings, make sure your program adds a null terminator prior to writing the text.</p> <p>FE_OK is returned if the operation succeeds.</p> <p>FE_IO_ADDR is returned if the register index is out of range.</p>
UINT16 GetCommError (UINT16 *data);	<p>GetCommError reads the communication error flag to the S register. The flag is a 1 bit integer value.</p> <p>FE_OK is returned always.</p>
UINT16 SetCommError (UINT16 data);	<p>SetCommError writes the communication error flag to the S register. The flag is a 1-bit integer value. You should only pass 0 or 1 to the SetCommError function. Using any other value can have unpredictable results.</p> <p>FE_OK is returned always.</p>
UINT16 GetAnalogAlarm (UINT16 index, INT16 *alm);	<p>GetAnalogAlarm reads an alarm status from an analog register indicated by 'index'. To learn more about available alarm statuses, refer to Understanding Alarm Statuses.</p> <p>NOTE: As of iFIX 4.5, only the alarm statuses IA_OK and IA_COMM are supported for use through the SM2 driver.</p> <p>FE_OK is returned if the operation succeeds.</p> <p>FE_IO_ADDR is returned if the register index is out of range.</p>
UINT16 SetAnalogAlarm (UINT16 index, INT16 alm);	<p>SetAnalogAlarm writes an alarm status to an analog register indicated by 'index' and causes an exception for the specified register even if the data or alarm has not changed. To learn more about available alarm statuses, refer to Understanding Alarm Statuses.</p> <p>NOTE: As of iFIX 4.5, only the alarm statuses IA_OK and IA_COMM are supported for use through the SM2 driver.</p> <p>FE_OK is returned if the operation succeeds.</p> <p>FE_IO_ADDR is returned if the register index is out of range.</p>
UINT16 GetDigitalAlarm (UINT16 index, INT16 *alm);	<p>GetDigitalAlarm reads an alarm status from a digital register indicated by 'index'. To learn more about available alarm statuses, refer to Understanding Alarm Statuses.</p> <p>NOTE: As of iFIX 4.5, only the alarm statuses IA_OK and IA_COMM are supported for use through the SM2 driver.</p> <p>FE_OK is returned if the operation succeeds.</p> <p>FE_IO_ADDR is returned if the register index is out of range.</p>
UINT16 SetDigitalAlarm (UINT16 index, INT16 alm);	<p>SetDigitalAlarm writes an alarm status to a digital register indicated by 'index' and causes an exception even if the data or alarm has not changed. The same alarm is associated with all 16 bits in the digital register. To learn more about available alarm statuses, refer to Understanding Alarm Statuses.</p> <p>NOTE: As of iFIX 4.5, only the alarm statuses IA_OK and IA_COMM are supported for use through the SM2 driver.</p> <p>FE_OK is returned if the operation succeeds.</p> <p>FE_IO_ADDR is returned if the register index is out of range.</p>

UINT16 GetTextAlarm(UINT16 index, INT16 alm)	<p>GetTextAlarm reads an alarm status from a text register indicated by 'index'. To learn more about available alarm statuses, refer to Understanding Alarm Statuses.</p> <p>NOTE: As of iFIX 4.5, only the alarm statuses IA_OK and IA_COMM are supported for use through the SM2 driver.</p> <p>FE_OK is returned if the operation succeeds.</p> <p>FE_IO_ADDR is returned if the register index is out of range.</p>
UINT16 SetTextAlarm(UINT16 index, INT16 alm)	<p>SetTextAlarm writes an alarm status to a text register indicated by 'index'. When a block reads data from the SM2 driver, the alarm status of the first byte is returned. The status of additional bytes is ignored. To learn more about available alarm statuses, refer to Understanding Alarm Statuses.</p> <p>NOTE: As of iFIX 4.5, only the alarm statuses IA_OK and IA_COMM are supported for use through the SM2 driver.</p> <p>FE_OK is returned if the operation succeeds.</p> <p>FE_IO_ADDR is returned if the register index is out of range.</p>

Understanding Alarm Statuses

iFIX can process alarm status information from I/O drivers. This information complements the alarms generated by iFIX data-base blocks. When an alarm is returned from a driver, iFIX compares the driver alarm against the block alarm. The alarm with the higher severity is used as the block alarm and the other alarm is ignored.

NOTE: As of iFIX 4.5, only the alarm statuses IA_OK and IA_COMM are supported for use through the SM2 driver.

iFIX defines the following alarms with the following severity:

Severity	Alarm Status	Description
16 (highest)	IA_COMM	Communication error ("BAD" value).
16 (highest)	IA_IOF	General I/O failure.
16 (highest)	IA_OCD	Open circuit.
16 (highest)	IA_URNG	Under range (clamped at 0).
16 (highest)	IA_ORNG	Over range (clamped at MAX).
16 (highest)	IA_RANG	Out of range (value unknown).
16 (highest)	IA_DEVICE	Device failure.
16 (highest)	IA_STATION	Station failure.
16 (highest)	IA_ACCESS	Access denied (privilege).
16 (highest)	IA_NODATA	On poll, but no data yet.
16 (highest)	IA_NOXDATA	Exception item, but no data yet.
16 (highest)	IA_MANL	Special code for MANL/MAINT (for inputs).
8	IA_FLT	Floating point error.
8	IA_ERROR	General block error.
8	IA_ANY	Any block alarm.
8	IA_NEW	New block alarm.
7	IA_HIHI	The block is in the HIHI alarm state (High High).

7	IA_LOLO	The block is in the LOLO alarm state (Low Low).
7	IA_COS	Change of state.
7	IA_CFN	Change From Normal (Digital block only).
7	IA_TIME	Time-out alarm.
7	IA_SQL_LOG	Not connected to database.
6	IA_HI	The block is in the HI alarm state (High).
6	IA_LO	The block is in the LO alarm state (Low).
6	IA_RATE	Value exceeds rate of change setting since last scan period.
6	IA_SQL_CMD	SQL command not found or invalid.
5	IA_DEV	Deviation from the set point.
5	IA_DATA_MATCH	SQL command does not match data list.
4	IA_FIELD_READ	Error reading tag values.
4	IA_FIELD_WRITE	Error writing tag values.
1	IA_DSAB	Alarms disabled.
0 (lowest)	IA_OK	The block is in normal state.

Using the preceding table, you can see that if a driver returns a HHHI alarm to a block that is in HI alarm, iFIX changes the alarm state to HHHI because the driver alarm is more severe. However, if the alarms are of equal severity, iFIX does not change the alarm state of the block. For example, if the block is in HI alarm and the driver returns a LO alarm, the block's alarm state does not change because both alarms have equal severity. Once an operator acknowledges the HI alarm, iFIX changes the block's alarm state.

NOTE: If you set a communication error to the S register with the SetCommError function, then all SM2 registers show a COMM alarm status. When examining the alarm status of text, only the status of the first character (byte) is read. You can control the alarm status functions of the SM2 driver using its C API only. Refer more information about this API, refer to the [Using the SM2 C API](#) section.

Connecting to an OPC Server

In addition to I/O drivers, the WorkSpace can send and receive data with an OLE for Process Control (OPC) server. You can configure any database block to receive or send OPC data by completing the block's I/O driver fields. To do this, select OPC from the Driver field, and then click the I/O Address field's Browse button to specify an OPC address.

The OPC address has the following syntax:

```
ServerName;GroupName;ItemID;AccessPath
```

where *ServerName* is the name of your OPC server, *GroupName* is the name of the OPC group you want to access, and *ItemID* is the name of the OPC item you want to read or write. Including the *AccessPath* is optional and instructs the server how to access its data. For more information about connecting to your OPC server, refer to your OPC PowerTool documentation.

Understanding Signal Conditioning

Very often raw values from your process hardware are not meaningful to operators. This is particularly true when the hardware reports values in a numeric format, such as an unsigned integer, to indicate how full a tank is. In this situation, what is needed is

a way to map the range of values you receive into a different range of values. Many I/O drivers provide this ability by applying signal conditioning.

Signal conditioning converts the data received from the process hardware into a format that is easily recognizable by operators. You can apply signal conditioning by selecting the type you want to use from a block's Signal Conditioning field.

Example: Understanding Signal Conditioning

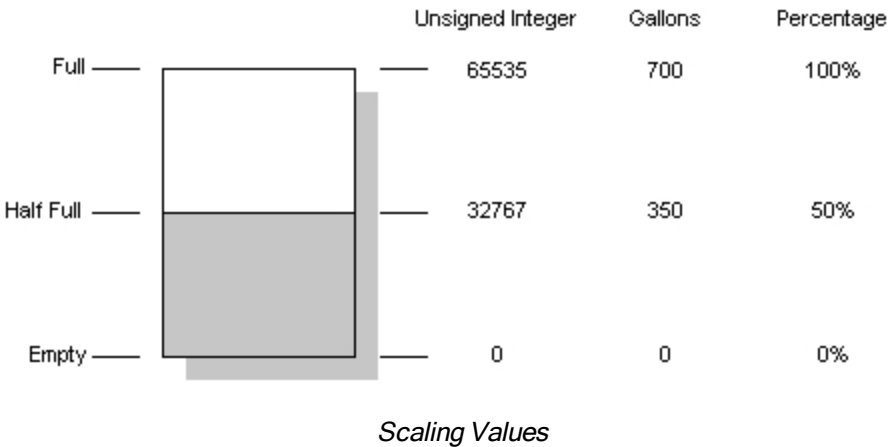
Suppose you have a 700-gallon water tank and you want to display how much water is in the tank. You can display the tank's water level as:

- Unscaled integer received from your process hardware
- Gallons
- Percent filled

For this example, assume the I/O driver sends an unsigned integer to the process database (that is, 0 to 65535). The following table lists sample high and low EGU limits you could assign to the input block. These settings scale the incoming values to display the tank's water level in percent filled and in gallons.

Example Block EGU Limits	
Operator Display: Limit Settings:	
Unsigned Integer	0 to 65535
Percent	0 to 100
Gallons	0 to 700

The following figure shows the values that are displayed when the tank is full, half full, and empty.



Understanding EGU Limits

EGU limits have a precision and range. The *precision* is the number of digits after the decimal point. The *range* is the span of values. For example, the default range for blocks is 0 to 100.

Changing the EGU Limit Precision

You can change the EGU limits' precision by editing the High Limit and Low Limit fields. When you change the precision, Database Manager modifies all references to the current block throughout the database. For example, if you create an Analog Input block with EGU limits of 0.0 to 100.0 and then change the precision to 0.00 to 100.00, Database Manager searches the database for all references to this block and makes the appropriate changes. In this case, a Program block that contains the following step:

```
SETOUT AI1 50.0
```

is adjusted to read:

```
SETOUT AI1 50.00
```

Changing the EGU Limit Range

To change the EGU limits' range, you must change all references to this block manually. For example, if you create an Analog Input block with limits of 0.0 to 100.0 and then change the range to 0.0 to 700.0, all references to this block's EGU limits are unaffected until you edit them. In this case, assume you have a Program block that outputs a value equal to half of this block's original range, as shown below:

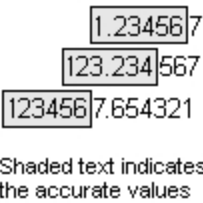
```
SETOUT AI1 50.0
```

You must modify this Program block's SETOUT statement to reflect the new range, as shown below:

```
SETOUT AI1 350.0
```

EGU Limit Formats

Database Manager accommodates the EGU limit formats listed in the table *Available EGU Limits*. Each format is accurate to six digits. Because of compiler limitations, round-off errors may occur in the seventh digit. The following figure shows sample values and their accuracy.



Available EGU Limits

Format:	Accepts Limits From:
Standard	-32768 to 32767 (signed int)
Integer	0 to 65535 (unsigned int)
	0 to 999 (3BCD)
	0 to 4095 (12 Binary)
Expanded	-9999999 to 9999999
Decimal Notation	Using this format, you can specify up to six places after the decimal point. Make sure you enter the same number of decimal places in fields that require both high and low limits. Note that this range would be -1E7 to +1E7 in scientific notation.
Scientific Notation	±3.4E-38 to ±3.4E+38
	Use this format to display large or small numbers. Again, only the first seven digits are accurate. If you prefer, you can also use scientific notation for numbers that iFIX can display in decimal notation. Using decimal notation with scientific notation, you can specify up to six places after the decimal point.

NOTE: Refer to your I/O driver manual for additional integer ranges supported by your equipment.

Working With the Process Database

As you develop your process databases, you need to complete many basic operations including:

- [Creating a new database](#)
- [Opening and closing an existing database](#)
- [Adding blocks to the current database](#)
- [Copying](#), [modifying](#), [duplicating](#), [displaying](#), and [deleting](#) database blocks
- [Moving blocks from one database to another](#)
- [Saving the current database](#)

This chapter describes how to complete these tasks.

Creating a New Database

One of your first tasks when developing a process database is to create a new database. Creating a new database loads the empty database, EMPTY.PDB, into memory and removes any existing database that may be loaded. If the existing database has unsaved changes, Database Manager prompts you to save them before the new database loads.

Opening and Closing a Database

Before Database Manager can open and display a database, the program establishes a connection to a SCADA server (either local or remote) on the network. It accomplishes this by prompting you to select the SCADA server to which you want to connect. Once you select the server, Database Manager establishes the connection and opens the server's current database.

You can establish a connection with a different server by opening a different database. Once you select the new server you want to connect to, Database Manager disconnects from the existing server and opens the selected server's current database.

NOTE: From the Database Manager, you can open databases from SCADA nodes with iFIX 3.0 or greater installed. You cannot add, modify, or delete blocks in databases from earlier versions of iFIX, such as iFIX 2.5 or iFIX32.

You can also disconnect from a SCADA server by closing the database. If the database has unsaved changes, Database Manager prompts you to save them before breaking the connection.

Adding Blocks

After you open or create a new database, you can begin adding blocks to your database. Database Manager lets you add blocks by:

- Double-clicking a blank cell in the spreadsheet.
- Right-clicking a cell in the spreadsheet and selecting Add block from the pop-up menu.

- Clicking the Add button from Database Manager's toolbar (Classic view).
- Clicking Add in the Blocks group on the Home tab (Ribbon view).
- Entering the name of a nonexistent block in the Next field of a block dialog box and clicking the arrow button. Refer to the [VBA Naming Conventions](#) section of the Writing Scripts manual for information on appropriate names.
- Using the Generate Wizard.

When any of the first four methods are used, you can select the type of block you want to add. Once you do this, a block configuration dialog box appears. By completing the dialog box and clicking OK, you can add the block to the database. For more information on completing a block configuration dialog box, refer to the [iFIX Database Reference](#) help.

NOTE: Whenever you add or delete a block in the database, we recommend you resolve your pictures. For more information about resolving pictures, refer to the [Creating Pictures](#) manual.

Adding Multiple Blocks

The Generate Wizard lets you add many similar blocks to the database quickly, saving development time. You simply select the type of block you want to create and the block names you want to use.

For the block type, you can select an existing or a new block. This selection determines the specific values assigned to each block created. For example, when you select a new block, the Wizard creates blocks with default values. However, when you select an existing block, the Wizard creates blocks with values of the selected block. This option lets you create many similar blocks quickly without having to reconfigure them later.

When specifying the block names, you must enter a prefix, suffix, starting number, ending number, and increment value. The Generate Wizard uses this information to systematically assign names to the blocks it creates. For example, the following values create blocks with the names F1T1 through F20T1.

Prefix	Starting number	Ending number	Increment	Suffix
F	1	20	1	T1

The Generate Wizard cannot create a block with a name that already exists. If the information you specify results in an existing block name, the Wizard skips that block and continues on to the next one.

Using the Generate Wizard, you also have the option of customizing up to 5 fields. Customizing these fields lets you fine tune the block's configuration. For example, if you need to create 50 Digital Input blocks, each one will have a different I/O address. You can use the Generate Wizard to assign these addresses if they occur in a sequential order.

You can customize a field by selecting it and entering a prefix, suffix, starting number, ending number, and increment value. The Wizard handles this information identically to the block name values you entered. If the resulting sequence of field values ends before the Wizard creates all the new blocks, the sequence repeats from the beginning. For example, suppose you want to create 10 Analog Input blocks, AI1 through AI10 and you customize the I/O address of each block with the following information:

Prefix	Starting number	Ending number	Increment	Suffix
N	10	50	10	:7

The blocks receive the following addresses:

Block	Address
-------	---------

AI1	N10:7
AI2	N20:7
AI3	N30:7
AI4	N40:7
AI5	N50:7
AI6	N10:7
AI7	N20:7
AI8	N30:7
AI9	N40:7
AI10	N50:7

You also have the option of enabling the Use Custom Format check box. When you enable this check box, the Wizard lets you enter multiple patterns for the five fields you have selected. A *pattern* acts as a programming statement for generating a range of numeric or string values. The following table lists the syntax for each type of pattern.

Pattern Types

Type	Syntax	Example
Numeric	<code><start:end:increment></code> where <i>start</i> is the initial value of the pattern, <i>end</i> is the ending value, and <i>increment</i> is the amount to add to the current pattern value.	<code><1:10:1></code> generates a range of numbers from 1 to 10. <code><1:20:3></code> generates the numbers 1,4,7,10,13,16, and 19.
Alphanumeric List	<code><"string1", "string2",... "string"></code> where <i>string1</i> is the first string, <i>string2</i> is the second string, and <i>string</i> is the last string. Each string must be enclosed in quotation marks.	<code><"A", "B", "C", "D"></code> <code><"Area A", "Area B", "Area C"></code> <code><"Pump 1", "Pump 2", "Pump 3"></code>
Literals	<code>string</code> where <i>string</i> is up to 40 alphanumeric characters.	Alarms Elapsed Time Analog Input
Constants	<code>"string"</code> where <i>string</i> is up to 40 alphanumeric characters, including the quotation marks.	<code>"Curr Value > 100"</code> <code>"I/O Addr < 500"</code> <code>"PLC:N7:"<766:780:1>"/15"</code>

Notice that the difference between literals and constants is minor. Both are strings and can be up 40 characters. However, quotation marks (" ") are required for constants when the string contains any mark of punctuation or non-alphanumeric symbol such as an angle bracket or a colon. If the string contains only numbers or letters, the quotation marks are optional.

Also notice that negative and floating point numbers are not supported within patterns. You can generate these types of numbers by enclosing a minus sign or a decimal point in quotation marks outside of the pattern. Negative increments are not supported. Consider the following examples:

The pattern...	Yields the result...
"-"<1:10:1>	-1 -2 -3 -4 -5 -6 -7 -8 -9 -10
<1:5:1>"."<"5">	1.5 2.5 3.5 4.5 5.5

By combining different pattern types, you can generate a wide range of block field entries. For example, if you want to customize the Description field for a group of blocks, you could enter a pattern such as Alarm Status from Area <"A", "B", "C">. Notice that the text outside the pattern does *not* require quotation marks.

Duplicating Blocks

Another way you can add similar blocks to the database is by duplicating them. Like the Generate Wizard, you can duplicate multiple blocks. However, you cannot customize specific block fields; you can only specify a new name for the duplicated blocks. As a result, the new blocks are identical to the original ones.

For example, suppose you have an Analog Input block monitoring the speed of a pump and want to monitor five other pumps in a similar way. By duplicating the block, you can reproduce it and create the additional blocks you need.

Copying and Pasting Blocks

Database Manager lets you share block information among your process databases and third-party applications by copying and pasting them to the clipboard. Copying and pasting blocks between two databases is a quick way to create new ones. This works best when the database you want to create contains blocks similar to existing blocks in other databases. By copying existing blocks, you save time because you only need to change the information that differs between the original block and the new copy.

Copying and pasting between Database Manager and a third-party application lets you quickly modify a block from your text editor or spreadsheet. You can also create new blocks with your third-party application and paste them into Database Manager. This feature can be helpful when you do not have access to iFIX and you want to create new blocks for later inclusion into the process database.

When you copy blocks to the clipboard, Database Manager converts and saves them in comma separated value (CSV) format. Once saved on the clipboard, you can paste the blocks into any application that supports CSV files.

Pasting blocks into Excel requires you to configure the program so that it converts and displays the data correctly. You also need to configure Excel when you drag and drop data from Database Manager or import a database saved in CSV format.

► **To configure Excel:**

1. Paste your blocks into an Excel spreadsheet.
2. On the Data menu, click Text to Columns.
3. Select Delimited.

4. Click Next.
5. Select the Comma check box.
6. Click Finish.

Pasting CSV-based blocks into Database Manager converts and adds them to the process database in memory. If the names of these blocks are already in use, Database Manager prompts you to replace the existing blocks with the ones you are pasting. If you do not replace them, Database Manager generates an error for each duplicate block. You can avoid creating these errors by loading a database that does not contain the blocks you are pasting or by modifying the names of the blocks prior to pasting them into Database Manager.

Database Manager also displays errors if you attempt to paste a block from Excel that is not in the correct format. The easiest way to ensure the block is in the correct format is to export a block of that type from the process database or paste a block into Excel, modify it, and paste the modified block back into Database Manager.

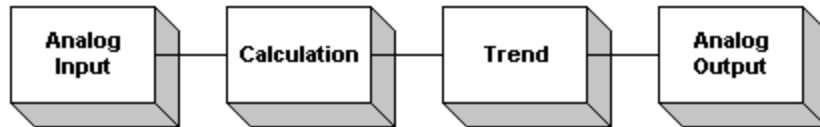
Moving Blocks

In addition to sharing blocks by copying and pasting them, you can cut and paste them. Cutting blocks from the database is similar to copying them; the main difference is that cut blocks are physically moved and placed on the clipboard, allowing you to move blocks to other databases.

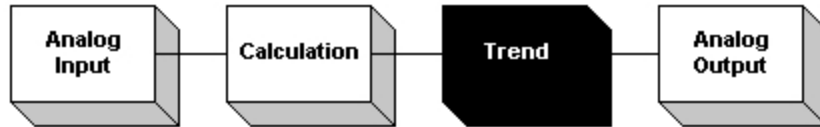
Prior to removing blocks from their chains, Database Manager takes each chain off scan. Make sure you place the chain on scan when you finish editing it. iFIX provides several ways to do this. For more information about placing a block or chain on scan, refer to the section [Placing Blocks On and Off Scan](#).

If you cut a block from the middle of the chain, Database Manager attempts to connect the two portions of the chain, as the following figure shows.

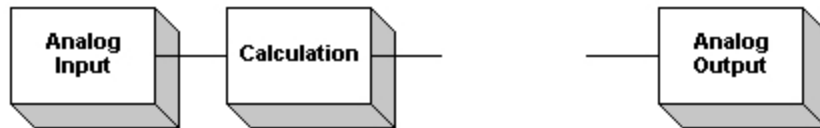
Assume you want to remove the Trend block from this chain.



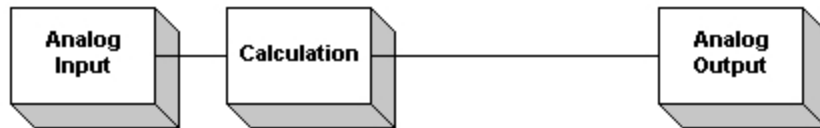
Select the block.



While the block is highlighted, select Cut or Delete to remove the block from the chain.



Once the block is removed, Database Manager fills in the hole and connects the Calculation and Analog Output blocks.



Removing a Block from a Chain

Modifying Blocks

You can modify any database block displayed in the spreadsheet. Typically, you need to modify a block:

- After you copy it into the database.
- If you discover an incorrect field value.
- Whenever your database needs change.

NOTE: While you can modify Analog Register and Digital Register blocks, we do not recommend that you modify the current value of these blocks using Database Manager. To learn more about using Analog Register and Digital Register blocks, refer to the section [Working with Analog and Digital Blocks](#).

Displaying Block Information

Before you modify a block, you can determine the exact fields to change by displaying the block's configuration dialog box. Unlike modifying a block, displaying a block's dialog box leaves the block and its chain on scan.

Deleting Blocks

Whenever you find that specific blocks are no longer needed, you can delete them from the database. Database Manager lets you delete blocks by selecting:

- The blocks you want to remove and cutting them without subsequently pasting them
- The Delete button from Database Manager's toolbar (Classic view)
- The Delete button from the Blocks group on the Home tab (Ribbon View).

The difference between these methods is the Delete button in Classic View or Delete in Ribbon view removes the selected blocks completely and does not let you retrieve them if you change your mind. By cutting blocks, you instruct Database Manager to save a copy on the clipboard allowing you to paste them back into the database until the next copy or cut occurs.

Regardless of the method you choose to remove the blocks, Database Manager displays a message box with the following text:

```
Following tag(s) selected for deletion from database
```

Click Cancel to retain the blocks and continue using Database Manager. If you click Delete All, Database Manager deletes the blocks and takes the chains containing them off scan. Make sure you place the chain on scan when you finish editing it. iFIX provides several ways to do this. For more information about placing a block or chain on scan, refer to the section [Placing Blocks On and Off Scan](#).

If the deleted block was in the middle of the chain, Database Manager attempts to connect the two portions of the chain as the [Removing a Block from a Chain](#) figure shows.

Saving a Database

When you finish making changes to a database, you can save the database to disk. By saving the database, you enable the SCADA server to reload the database in memory when you restart iFIX.

Make sure you have enough disk space available when saving your database to disk. If your SCADA server does not have enough disk space, you may lose the changes you have just made. For example, if you build a 3MB database, make sure you have 3MB of free disk space available.

Saving a Database from a Script or a Program Block

You can also save a database automatically using the RUNTASK command from a Program block or a script. When you save a database using the Program block, use the following syntax:

```
RUNTASK DBBSAVE -Nnodename -Ddatabase
```

The -N command line parameter enables you to save a database on a remote SCADA server. The -D command line parameter enables you to save a database to another PDB file name. Both command line parameters are optional; if you do not use them, you save the current database on the local SCADA server.

When you save a database from a VBA script, use routines similar to the following:

```
Private Sub FixEvent1_OnTrue()  
Dim save_pdb As String
```

```

Dim return_value As Double
Dim nodename As String
Dim database As String

nodename = "MIXER1"
database = "BACKUP"
save_pdb = System.ProjectPath + "\DBBSAVE" + " -N" + nodename + " -D" + database
return_value = Shell (save_pdb, 0)
End Sub

```

End SubRefer to the Database Manager online help system for more information about the RUNTASK from the Program block. For information about the Shell function, refer to the iFIX Automation Interfaces Help file.

Locating and Displaying Data

Using Database Manager, you can locate and display data in the spreadsheet. This chapter explains how to accomplish these tasks. Refer to the following topics for detailed information:

- [Finding Data in a Spreadsheet](#)
- [Replacing data](#)
- [Using Go To](#)
- [Updating and Pausing the Spreadsheet](#)
- [Customizing the Spreadsheet](#)
- [Saving and Loading Spreadsheet Layouts](#)
- [Overriding the Default Layout](#)

Finding Data in a Spreadsheet

As you develop and test your process database, you may find it necessary to locate specific spreadsheet values. For example, you may want to locate and adjust the scan times of your primary blocks.

With Database Manager, you can search in any spreadsheet column for text by selecting the column head of the column you want to search in. You must also enter a *search string*. The string is the text you want to locate in the selected column. You can enter up to 29 alphanumeric characters for the search string.

Once you enter the text you want to locate, Database Manager locates the first occurrence of the string in the selected column. To locate subsequent occurrences, you can repeat this process.

Find Options

You can toggle the case sensitivity of a search using the Match Case option. When this option is enabled, Database Manager searches for the exact text and case you enter. The search string is *not* case sensitive when you disable the option. For example, selecting the Tag Name column and entering the search string:

is the same as entering any of the following search strings:

- ai
- Ai
- al

You can also find whole or partial words with the Match Whole Word Only option. Database Manager treats the search string as a whole word when you enable the option. By disabling the option, you can find the specified string inside other words. For example, if you search for the string "line" with the Match Whole Word Only option disabled, Database Manager matches the following descriptions with the search string:

- Alarms for Line 1
- Alkaline Pump
- Linear accelerator for proton chamber

Replacing Data

In addition to finding text, you can find and replace it. Finding and replacing text is similar to just finding it. Both tasks require you to select the cell or column you want to search in and enter a search string. However, when finding and replacing data, you can also enter a *replacement string*. This text is the data with which you want to replace the search string.

Find Options

You can enter up to 29 alphanumeric characters for both the search and replacement strings. Both strings also accept the Match Case and Match Whole Words Only options. However, while you can toggle the case sensitivity of the search string with the Match Case option, the replacement string is always case sensitive. This means that when replacing data, Database Manager inserts text exactly as you enter it.

You also have the option to replace the text in the current selection (a cell or column) or the current column. The Selection option button lets you replace text in the current selection. The Entire Column option button lets you replace text in the entire column.

Using Go To

While working with your database, you may want to display the text in a specific row or column. When the number of rows or columns is small, using the scroll bars to move through the spreadsheet is a quick way to display the necessary information.

However, as the spreadsheet grows, you may find scrolling through it slow and time-consuming. To speed up locating information, you can jump to any row or column. You can also jump to any block in the spreadsheet when do not know the block's row number by entering its name instead.

Updating and Pausing the Spreadsheet

You can configure Database Manager to automatically update the spreadsheet by enabling the Auto Refresh option. This option periodically refreshes the values in the spreadsheet. You can enter refresh rates between 5 and 3600 seconds.

By automatically updating the spreadsheet, you can troubleshoot your database by monitoring specific blocks. Should you identify an error, you can correct it and immediately see your change's effect.

You can temporarily disable the automatic refresh option by pausing the spreadsheet. Pausing updates instructs the Database Manager to stop updating the screen. It does not affect SAC or the scan status of the blocks in the database.

NOTE: Database Manager automatically pauses updating the screen when you add, modify, delete, or generate blocks or when you print, reload, or import the database.

In addition to automatically refreshing the screen, you can manually update it whenever you want to refresh the values in the spreadsheet. Manually updating the screen can be particularly helpful when you automatically update the spreadsheet infrequently, and can be used in conjunction with the automatic refresh option to update the screen more frequently than every 5 seconds.

Customizing the Spreadsheet

Database Manager lets you customize the spreadsheet by:

- [Changing its colors.](#)
- [Defining a non-scrolling column.](#)
- [Defining a spreadsheet layout.](#)

By completing these tasks, you can create a custom display for yourself and other database developers.

Coloring the Spreadsheet

Changing spreadsheet colors is like changing the colors of a window with the Display dialog box in the Control Panel. First, you select the item whose color you want to change and then you select the color for it.

You can change the color of the following spreadsheet properties:

- Cell text
- Spreadsheet grid
- Row and column head background and text

Creating a Non-Scrolling Column

As its name suggests, a non-scrolling column is one that does not move when you scroll left or right, as the following figure shows.

Spreadsheet before scrolling

	Tag Name	Type	Scan Time
1			
2			
3			
4			
5			

Non-scrolling column

Spreadsheet after scrolling

	Tag Name	Type	Scan Time
1			
2			
3			
4			
5			

Defining a Non-Scrolling Column

Using this feature, you can display related information without resizing or rearranging the spreadsheet columns. For example, if you configure the Type column to stop scrolling, you can easily relate block values, names, and types, allowing you to identify the block containing the values in each column.

When you create a non-scrolling column, all columns to the left of it no longer scroll until you unlock them.

Defining a Spreadsheet Layout

You can define a spreadsheet layout by selecting the:

- Columns you want to display.
- Order these columns appear in the spreadsheet.
- Column headings.
- Columns' width.

The columns you can select correspond to common and block-specific fields. Columns corresponding to common block fields appear with only the field name. For example, the following text appears for the common field Current Value:

`Curr Value`

Columns corresponding to block-specific fields use the following naming convention:

`typefieldname`

The *type* is a two or three-character abbreviation for the block type and the *fieldname* is the name of the block field. For example, suppose you see the following text:

AI Smooth

AI is the block abbreviation and identifies the field as an Analog Input specific field. The text Smooth indicates the field name and corresponds to a block field (in this case the A_SMOTH field). The [Block Type Abbreviations](#) table lists the abbreviations for each block.

NOTE: If you remove a column from the spreadsheet that Database Manager uses to sort data, the column is also removed from the sort order.

Saving and Loading Spreadsheet Layouts

If you display database information in different ways, you may want to save your spreadsheet layouts to a *format file*. Format files define the following information for each spreadsheet column:

- The column heading.
- The column width.
- The block field to which the column corresponds.

By saving spreadsheet layouts, you eliminate the need to recreate them. Database Manager saves your spreadsheet layout in format files residing in the FIX Local path by default. These files have the extension .FMT.

Loading a format file is like creating a spreadsheet layout. It appears in the Properties dialog box, allowing you to modify it as needed. In addition, when Database Manager opens the format file, the program replaces the current set of spreadsheet columns with the ones listed in the format file. If a block does not include a field that corresponds to one of the new columns, the text "-----" appears in that cell.

Overriding the Default Layout

The first time you run Database Manager, it loads its preset default spreadsheet layout stored in the format file, DEFAULT.FMT. The following table lists the preset default columns and the corresponding fields that appear.

Database Manager Default Layout	
Default Column	Corresponding Field
Tag Name	A_TAG
Type	A_NAME
Description	A_DESC
Scan Time	A_SCANT
I/O Dev	A_IODV
I/O Addr	A_IOAD
Curr Value	A_CV

If you prefer a different default arrangement of the spreadsheet columns, you can override the preset default layout with your own by saving the current settings. You can also restore the original layout as the default by loading the format file and selecting Save Settings from the Tools menu.

Troubleshooting Tag Display

If the Database Manager displays tag values, but the fields for an open block appear empty in the block display, try re-registering the Fixdb32egu.ocx, Fixdb32IOAddress.ocx, and Fixdb32Tagname.ocx files. This should resolve the issue with the tag display. Use the steps below.

► To resolve tag display problems:

1. Shut down the iFIX Database Manager.
2. Shut down the iFIX WorkSpace.
3. From the Start menu, click Run. The Run dialog box appears.
4. In the Open field, enter:
`REGSVR32 -U "C:\Program Files (x86)\GE\iFIX\Fixdb32egu.ocx"`
5. Click OK. A message should appear indicating that the "DllUnregisterserver" succeeded.
6. Click OK.
7. Repeat steps 3-6 to properly unregister these files:
 - Fixdb32IOAddress.ocx
 - Fixdb32Tagname.ocx (making the appropriate substitution for the file name)
8. From the Start menu, click Run. The Run dialog box appears.
9. In the Open field, enter:
`REGSVR32 "C:\Program Files (x86)\GE\iFIX\Fixdb32egu.ocx"`
10. Click OK. A message should appear indicating that the "DllRegisterserver" succeeded.
11. Click OK.
12. Repeat steps 8-11 to properly register these files:
 - Fixdb32IOAddress.ocx
 - Fixdb32Tagname.ocx (making the appropriate substitution for the file name)
13. Restart the iFIX WorkSpace.
14. Start the iFIX Database Manager, open a tag, and confirm all of the proper tag information appears.

Querying and Sorting Data

One of the most powerful features that Database Manager provides is querying a database. A *query* is a request to display specific information. For example, you might want to display all the Analog Input blocks with a scan time of five seconds or less. Alternatively, you might want to display every block in security area Packaging for a specific device driver. After you create a query, Database Manager selects the blocks from the database that match the request and displays them, replacing any previously displayed blocks.

The following topics provide more detailed information about querying and sorting a database.

- [Understanding Query Syntax](#)
- [Editing a Query](#)
- [Refining and Expanding a Query](#)
- [Saving and Loading a Query](#)
- [Overriding the Default Query](#)
- [Understanding Sort Orders](#)
- [Saving and Loading a Sort Order](#)
- [Changing the Default Sort Order](#)

Understanding Query Syntax

Before you can create a query, you must know the information you want to retrieve from the database. For example, you might want to display all the Analog Input blocks with a current value greater than 50. Alternatively, you might want to display every block in alarm area Line5 for a specific I/O driver.

Once you know the information you want to display, you can construct the query using the following syntax:

```
{column} operator "value"
```

or

```
"value" operator {column}
```

where *column* is a column heading you want to display, enclosed in curved brackets, *operator* is one of the relational operators listed in the table *Relational Operators*, and *value* is a number or string enclosed with quotation marks (" ").

Relational Operators	
Operator	Meaning
=	Equal to
<=	Less than or equal to
<	Less than
!=	Not equal to
>=	Greater than or equal to
>	Greater than

Case Sensitivity

Strings entered in a query are not case sensitive. For example, a query such as:

```
{Tag Name} = "AI*"
```

retrieves blocks with names that start with AI. It also retrieves blocks with names that start with:

- Ai
- ai
- al

When entering a number, the block's EGU range determines the maximum and minimum values you can enter. The following table lists these limits. To learn more about the EGU range, refer to the section [Understanding EGU Limits](#).

Available EGU Limits	
The format...	Accepts EGU Limits From...
Standard Integer	-32768 to 32767 (signed int)
	0 to 65535 (unsigned int)
	0 to 999 (3BCD)
	0 to 4095 (12 Binary)
Expanded Decimal Notation	-9999999 to 9999999
Scientific Notation	+/-3.4E-38 to +/-3.4E+38

Each query you create is evaluated from left to right.

Using Boolean Operators

You can combine two or more queries using one of the boolean operators in the following table. Using these operators you could build a query such as:

```
{Scan time} = "3" AND {Security Area 1} = "Packaging"
```

to display all the blocks in security area Packaging with a scan time of 3 seconds.

Boolean Operators		
The operator...	Instructs Database Manager to...	Example
AND	Select the block when both query A and query B are true.	A AND B
OR	Select the block if either query A or query B is true.	A OR B
NOT	Invert query. If query A is true, its value becomes false.	NOT A
NOR	Select the block when both query A and query B are false.	A NOR B
XOR	Select the block if query A and query B are not both true or both false.	A XOR B
NAND	Select the block when query A and query B are not both true.	A NAND B

Using Wildcards in a Query

You can also include the following wildcards in queries containing the equal to (=) operator.

The wildcard...	Represents...
Asterisk (*)	One or more characters.
Question mark (?)	Any single character.

Examples: Query Wildcards

The query...	Retrieves...
{Type} = ""	Every block in the database.
{Tag Name} = "A"	Every block whose name that begins with an "A."
{Type} = "?I"	All Analog Input and Digital Input blocks.

Other operators, such as the greater than (>) or less than (<) operators, treat wildcards literally.

Examples: Using Relational Operators

The query...	Retrieves...
{Scan Time} > "?M"	No blocks, because the question mark is treated literally and "1M" through "9M" have a lesser ASCII value than "?M".
{Scan Time} = "?M"	All blocks with a scan time from 1 minute to 9 minutes.

Grouping Queries

You can group queries together using parenthesis. For example, consider the following query:

```
{Tag name} = "A*" AND ({Type} = "AI" OR {Type} = "AO")
```

Database Manager evaluates this query from left to right. Expressions enclosed in parenthesis are treated as a unit. Consequently, as Database Manager evaluates the above query, it retrieves every block that begins with an "A." From this list of blocks, the program then displays all the Analog Input or Analog Output blocks.

Be careful where you place parenthesis in a query. The following query yields a very different spreadsheet compared to the query given above.

```
((Tag name) = "A*" AND {Type} = "AI") OR {Type} = "AO"
```

This query displays every Analog Input block that begins with an "A" and every Analog Output block in the database.

Editing a Query

After you create a query, it remains the current query until you change it. Using this feature, you can append a new query to the existing one and re-query the database. For example, suppose you create the following query:

```
{Type} = "AI" AND {Alarm Areas} = "Boston"
```

This query displays all the Analog Input blocks in the Boston alarm area. By appending the query:

```
AND {AI LOLO Alarm} <= "10"
```

you can display only those Analog Input blocks in the alarm area which have a LOLO alarm value less than or equal to ten.

► To locate blank cells in the database:

1. In Classic view, on the View menu, click Properties.

-Or-

In Ribbon view, on the View tab, in the Settings group, click Properties.

2. Select the Query tab and enter the following query.

```
{TAG NAME} = "*" AND {column_name}=""
```

For example, to find blank cells in the Description column use the query:

```
{TAG NAME} = "*" AND {Description}=""
```

This query displays all the blocks in the database that have no text in their Description fields.

Refining and Expanding a Query

The number of blocks that Database Manager retrieves from a query depends on how broad your request is. If the query is broad (for example, display all Analog Input blocks) the program retrieves more blocks than if the query is narrow (for example, display all Analog Input blocks with a scan time of five seconds and a name that begins with "Q"). You can adjust the number of blocks retrieved by refining or expanding your query.

One way to refine a query is to start with a simple query and append other queries to it. For example, the following query displays all the Analog Input blocks in the database.

```
{Type} = "AI"
```

Once Database Manager retrieves these blocks, you can refine the query by adding other queries to it. For example, you could add the following query to display only the Analog Input blocks with a scan time of two seconds:

```
AND {Scan Time} = "2"
```

Similarly, you could increase the number of blocks in the spreadsheet by displaying all the Analog Input blocks and then, to the original query, add all Analog Output blocks in the database with the following query:

```
OR {Type} = "AO"
```

Saving and Loading a Query

If you find yourself frequently entering and re-entering the same query, you may want to save it to a file. By saving a query, you cut down on frequent retyping and eliminate the possibility of mistyping the query. Database Manager saves your queries in separate files. These files reside in the FIX Local path by default and have the extension .QRY.

Loading a query is like entering one. It appears in the Enter Query field, allowing you to refine or expand the query as needed. In addition, when Database Manager opens the query file, the program retrieves the blocks that match the specified criteria.

Overriding the Default Query

By default, Database Manager loads the query file DEFAULT.QRY when you initially start the program. This query retrieves every block from the database. If you routinely request certain information from the database, you may want to override the current default query by saving the current settings. You can also restore the original query, DEFAULT.QRY, as the default, by loading the query file and selecting Save Settings from the Tools menu.

Understanding Sort Orders

Database Manager automatically sorts all the blocks in the spreadsheet whenever you query or open a database. The blocks are sorted according to the current *sort order*. A sort order defines:

- The columns to sort on.
- The column to sort first, the column to sort second, and so on.
- How to sort each column, either in ascending or descending order.

How Sort Orders Work

Database Manager sorts the spreadsheet, by:

1. Identifying which columns to sort.
2. Sorting the first column in ascending or descending order.
3. Repeating the previous step for the remaining columns in the sort order.

When sorting by ascending or descending order, Database Manager sorts:

- Special characters (such as marks of punctuation) by ASCII value.
- Numbers by numeric value.
- Letters in alphabetical order.

For example, suppose you want to sort the spreadsheet by the Type and Scan Time columns in ascending order. Once you select these columns:

- Database Manager sorts the spreadsheet by the Type column alphabetically (ascending order).
- Next, the program sorts the Scan Time column in numeric order.

Saving and Loading a Sort Order

If you sort your database in different ways, you may want to save each sort order to a file. By saving your sort orders, you eliminate the need to recreate it. Database Manager saves each sort order in a separate file. These files reside in the FIX Local path by default and have the extension .SRT.

Loading a sort order is like entering one. It appears in the Properties dialog box, allowing you to modify it as needed. In addition, when Database Manager opens the sort order file, the program re-sorts the spreadsheet.

Before loading a sort order, verify that the columns Database Manager sorts on appear in the spreadsheet. If the columns do not appear, add them. For more information about adding columns to the spreadsheet, refer to the chapter [Locating and Displaying Data](#).

NOTE: Because the Tag Name column only contains unique data, Database Manager never needs to sort beyond this column. For this reason, when you save or load a sort order containing the Tag Name column, Database Manager ignores any column that follows. Ignored columns are not saved or loaded.

Changing the Default Sort Order

By default, Database Manager loads the sort order DEFAULT.SRT when you initially start the program. This sort order arranges the blocks in the spreadsheet by block type and then by block name. If you prefer a different sort order, you can override the default by saving the current settings. You can also restore the original sort order, DEFAULT.SRT, as the default, by loading the sort order file and selecting Save Settings from the Tools menu.

Managing Databases

Database Manager lets you manage your process databases in many different ways. For example, you can verify them to ensure they do not contain any errors. You can also reload, import, and export your databases as needed. This chapter describes all these management tools.

- [Verifying Databases](#)
- [Reloading Databases](#)
- [Displaying a Database Summary](#)
- [Exporting Databases](#)
- [Importing Databases](#)

Verifying Databases

You can ensure a process database contains no configuration errors by verifying it. Verifying a database also ensures that iFIX can process each block and that the database functions as you intend.

While verifying a process database, Database Manager ensures that each block:

- Is in only one chain.
- Is linked to an appropriate block (for example, a Statistical Control block can be preceded only by a Statistical Data block).
- Is used in the correct context (for example, as a stand-alone, a primary, or a secondary block).
- Does not reference non-existent blocks.

If Database Manager detects no errors, it displays a message box to inform you. However, if it encounters errors, the Verify Database dialog box appears. This dialog box lists each error and the block that contains it. The following table lists the possible verification errors and how to resolve them.

Resolving Verification Messages		
When you see the message...	It means...	To correct this configuration...
<i>tagname a</i> : <i>tagname b</i> Tag is in more than one chain	The block, <i>tagname b</i> , has more than one upstream block linked to it. <i>Tagname a</i> identifies one of these blocks.	Remove one or more links to <i>tagname b</i> .
<i>tagname</i> :Tag is not in any chain	You may have a secondary block that is not in any chain or is the first block in a chain.	Remove the secondary block or add a primary block to the start of the chain.
<i>tagname</i> : Block not found for NEXT	The block, <i>tagname</i> , chains to a block that does not exist.	Create a block with the name specified in the Next field or enter the name of a block that exists.
<i>tagname</i> :Chains to itself	The block, <i>tagname</i> , contains its own name in its Next field.	Change the name in the Next field or leave it blank. If you want to repeatedly perform a task, use a Program block instead.
<i>tagname</i> : is not defined	The block, <i>tagname</i> , does not exist and another block references it.	Create the block or change the reference to a block that exists.
<i>fieldname</i> No such field in FDT	The field, <i>fieldname</i> , does not exist and it is referenced by a block in the database.	Change the reference to a field that exists.
Exceeding MAX chain size of 30	The database contains a chain with more than 30 blocks.	Redesign this chain by breaking it into two smaller chains or remove any unnecessary blocks.

Correcting Errors

You can correct any error by double-clicking it to display the associated block configuration dialog box.

► To correct errors:

1. Edit the block to correct the problem.
2. Save the block.
3. Re-verify the database.

Reloading Databases

Even though you can create multiple databases for a SCADA server, Database Manager can load and display only one of them at a time. You can load any database residing on the current SCADA server by selecting Reload from the Database menu. Reloading a database lets you:

- Switch from one database to another.
- Restore the database to its saved configuration.
- Load a database after completing and saving modifications.
- Place on scan the chains configured to begin processing when SAC starts.

Using the SCU, you can select the default database to load on start-up. When you load another database, it remains in memory until you restart iFIX. Refer to the [Setting up the Environment](#) manual for more information about specifying the database to load on start-up.

Reloading a Database from a Visual Basic Script

If you want operators to reload the database from the iFIX WorkSpace, you can create a script to reload the database using a routine similar to the following:

```
Private Sub FixEvent1_OnTrue()  
Dim load_pdb As String  
Dim return_value As Double  
Dim nodename As String  
Dim database As String  
  
nodename = "MIXER1"  
database = "BACKUP"  
load_pdb = System.ProjectPath + "\DBBLOAD" + " -N" + nodename + " -D" + database  
return_value = Shell (load_pdb, 0)  
End Sub
```

The -N command line parameter enables you to reload a database on a remote SCADA server. The -D command line parameter enables you to reload a database other than the currently loaded one. Both command line parameters are optional; if you do not use them, you reload the open database on the local SCADA server.

For information about using the Shell function from a script, see the Visual Basic help system.

CAUTION: The DBBLOAD system task does not prompt you to save changes to the database. Be sure you save your database before executing a script that reloads the database. Otherwise, you will lose all your changes.

Displaying a Database Summary

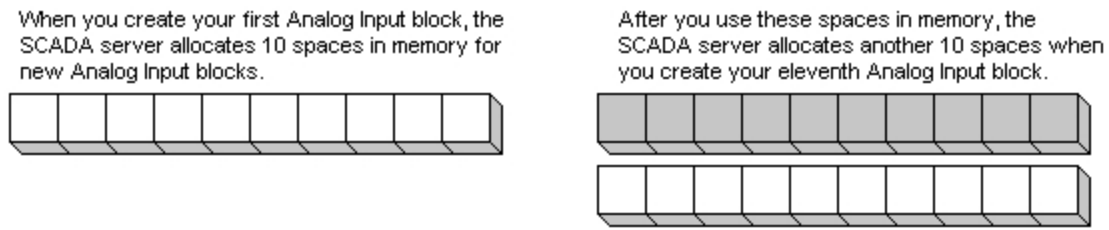
Database Manager lets you display a summary of the current database. This summary lists the database's:

- Size in bytes.
- Serial number.
- Contents.
- I/O count.

Using this information can help you manage your databases. For example, knowing the database's serial number can help you determine if someone modified your database since its last save. The serial number is a unique code that Database Manager creates whenever you add or delete a block. By writing down the serial number after you modify the database, you can subsequently compare the current number to the previous one.

High-level documentation about the database can also help you manage your databases. You can provide this level of information by displaying the database's contents. The contents lists by block type the number of blocks used and allocated. A used block is one you have configured. An allocated block is a placeholder in memory. To make efficient use of memory and improve

performance, all SCADA servers pre-allocate 10 blocks at a time when you initially create a block of a specific type, as the following figure shows.



Allocating Memory for Database Blocks

As an option, you can restrict your process database to a specific number of I/O blocks (intended for small applications that do not require many I/O points). In this environment, the I/O count lets you determine how many I/O blocks are in use.

Exporting Databases

You can export the current process database to a comma separated value (CSV) or GDB text file. Typically, you export a database when you want to:

- Complete large editing tasks using a text editor or spreadsheet,
- Change the alarm area database used by the process database; or
- Import it into a relational database for subsequent analysis.

For example, you could export your process database in CSV format and import the data into Microsoft Excel. By importing the file into Excel, you can:

- Automatically number or enter block values.
- Format values with Excel styles.
- Calculate a value with a spreadsheet formula.
- Automate the creation and modification of blocks by creating and running macros.

NOTE: If you export the iFIX Database into CSV format and open the CSV file in Microsoft Excel, you may notice that some Numeric fields containing decimals display without the decimal in Excel. For example, 100.01 displays as 100.01, however 100.00 displays as 100 in Excel. In order to display the entire number including the decimal (for instance: 100.00), update the cell format to a Numeric format.

For more information using these features, refer to your Excel documentation.

Database Manager exports all the blocks displayed in the spreadsheet. As a result, you can query the database to display only the blocks you want to export. For example, to export all the Analog Input blocks in the database, use the query:

```
{TYPE} = "AI"
```

For more information about querying the database, refer to the section [Understanding Query Syntax](#).

Exporting a Database from the Command Line

You can export a database from the command line using the DBExporter.exe command. This command is used independently of the Database Manager.

The DBExporter.exe command allows you to export all tags currently loaded on local or remote node to a comma separated value (CSV) or GDB text file. When no command parameters are specified, DBExporter.exe exports tags on local node to PDB name.csv to local PDB path. Typically, this path is: C:\Program Files (x86)\GE\iFIX\PDB.

Syntax

```
DBExporter.exe [/NNodeName] [/OOutputFileName>] [/F] [/R]
```

Parameters

The following table lists the command line parameters available for the DBExporter.exe exporter tool.

Parameter	Description
/NNodeName	Optionally, specify the remote SCADA node name that you want to to export the database from. Replace <i>NodeName</i> with the actual node name.
/OOut-putFileName	Optionally, specify the output file name with full path or no path. If no path is specified, output file is created in local PDB path. Typically, this path is: C:\Program Files (x86)\GE\iFIX\PDB. The file name portion requires the file extension: .csv or .gdb.
	Replace <i>OutputFileName</i> with the actual output file name. For example: MyFile.csv.
/F	Optionally, allows you to run a fast export. The export runs as fast as possible, maximizing CPU usage up to 100%.
/R	Optionally, allows you to skip header information (that includes the node name, database name, file name, and date and time of the export), in the output file.

Error Codes

The following table describes the error codes and descriptions DBExporter.exe exporter tool.

Code	Result
0	Successfully exported.
1	General Failure. iFIX is not running; unable to load DatabaseManagerRes.dll, an so on.
2	Invalid commands.
3	Invalid node.
4	Invalid file name.
5	Invalid file extension.
6	Unable to open Export.err file used for recording error details.
7	Empty database.
8	Invalid output file.
9	Exporting failure. The export file had an open or write error, EDA call failure, or other similar type error during the export process.

Editing the Export File

Once you export the database, you can edit it with any text editor. As you edit the export file, use the following guidelines:

- Verify that each block name is 256 characters or less.
- Ensure each block field is spelled correctly.
- Confirm the block type matches one of the types listed in the [Block Type Abbreviations](#) table.

You can also enter comments into the export file by typing an exclamation point (!) as the first character on a line.

CSV File Format

For more intricate editing of the export file, you need to understand the file format. For example, an export file saved in CSV format lists each row of the spreadsheet as one row in the CSV file. Each block's fields appear delimited by commas. The first two fields of each block are:

```
A_NAME, A_TAG
```

You only need to enter values for these two fields when creating new blocks. If no additional field values are present, Database Manager creates the block using the default block values. However, by adding these optional values to a block entry, you can complete specific dialog box fields when the block is imported. For example, the following line creates an Analog Input block with a scan time of 1 minute:

```
AI, AI2, , , , 1M
```

GDB File Format

The format of the GDB file is different. It uses the syntax:

```
label :: field_value ; fieldname
```

The first two lines of each block entry are required and contain the block type and block name. Without these lines, Database Manager will not add the block to the database when you subsequently import it.

Subsequent lines in a block entry appear for each field on the block's configuration dialog box. Including these additional lines is optional. If no additional lines are present, Database Manager creates the block using the default block values. However, by adding these optional lines to a block entry, you can complete specific dialog box fields when the block is imported. For example, to set the scan time of the block AI1 to 5 seconds, the export file should contain the following information:

```
Block Type:: AI ;A_NAME
Tag Name :: AI1 ;A_TAG

Scan Time :: 5 ; A_SCANT
```

Block Type Abbreviations

Block Type	Abbreviation
Analog Alarm	AA
Analog Input	AI
Analog Output	AO
Analog Register	AR
Boolean	BL
Calculation	CA
Dead Time	DT
Device Control	DC
Digital Alarm	DA
Digital Input	DI
Digital Output	DO
Digital Register	DR

Event Action	EV
Extended Trend	ETR
Fanout	FN
Histogram	HS
Lead Lag	LL
Multistate Digital Input	MDI
On-Off Control	BB
Pareto	PA
PID	PID
Program	PG
Ramp	RM
Ratio/Bias	RB
Signal Select	SS
SQL Data	SQD
SQL Trigger	SQT
Statistical Control	SC
Statistical Data	SD
Text	TX
Timer	TM
Totalizer	TT
Trend	TR

Importing Databases

Using Database Manager, you can import any database into memory. Importing a database merges it with the database currently in memory.

NOTE: Before you import a database, stop any I/O drivers or OPC servers you may have running.

Typically, you import a database when you want to:

- Change the scanning order of the blocks in the database.
- Combine two databases.

► To change the scanning order of a database's blocks:

1. Export a database. For more information about exporting a database, refer to the section [Exporting Databases](#).
2. Change the order in which the blocks are listed in the export file. For more information on a database's scanning order, refer to the section [Changing a Database's Scanning Order](#).
3. In Classic view, on Database Manager's toolbar, click the New button.
-Or-
In Ribbon view, click the Database Manager button, and then click New.
4. Import the edited export file.

This process ensures that only the original database resides in memory when Database Manager imports your database.

► To combine two process databases:

1. Export each database.
2. Examine each export file. Verify that each block in each database has a unique name. If two blocks have the same name, change one. Remember to also change the text in the Next field of the previous block so that the chain remains intact.
3. Into an empty database, import one of the databases you want to merge.
4. Import the other database.

Import errors encountered by Database Manager are stored in the file IMPORT.ERR. This file resides in the Database path.

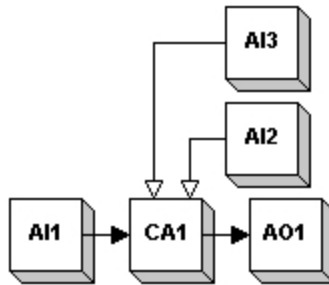
Advanced Topics

After you put a database into production, you may find that SAC scans certain blocks in an order you did not intend. This chapter explains how you can fine-tune your database and [change the scanning order](#).

This chapter also describes how to customize Database Manager's toolbar and Tools menu. Customizing the toolbar lets you select only the toolbar buttons you require. Customizing the Tools menu lets you add menu items that launch other applications from Database Manager. Refer to sections [Customizing the Toolbar](#) and [Customizing the Tools Menu](#) for more information on these features.

Changing a Database's Scanning Order

You can control the precise order you want SAC to scan your blocks and chains by adjusting the database's scanning order, called the *order of solve*. Typically, you only need to adjust the order of solve when the value of one block depends on the value of another. For example, suppose you are calculating a value from three Analog Input blocks as the following figure shows.



Sample Calculation Chain

Also assume that by default SAC scans AI1 first, AI2 second, and AI3 third. Because SAC scans AI1 first, it processes the entire chain, computing the value of the Calculation block, before it scans AI2 and AI3. If either of these values change during the current scan cycle, the Calculation does not receive them until AI1 is processed again.

Changing the scanning order addresses this potential problem. By scanning AI1 last, you ensure SAC has updated values for both AI2 and AI3 prior to computing the value of the Calculation block.

You can also phase the blocks so that SAC scans AI1 last. To learn more about phasing, refer to the [Phasing](#) section.

Understanding a Database's Scanning Order

On each scan cycle:

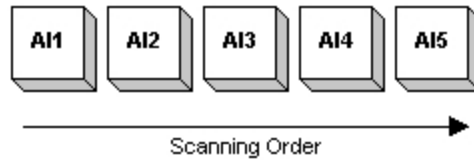
1. Regardless of their order in the database, SAC processes blocks that output a value to an I/O device (using a Cold Start value). SAC does not examine these values for alarm conditions.
2. SAC updates all primary blocks according to the scan time and phase. Blocks with the same scan time and phase update in the following order.
 - a. Analog Input blocks
 - b. Analog Output blocks
 - c. Digital Input blocks
 - d. Digital Output blocks
 - e. Ramp blocks
 - f. Multistate Digital Input blocks
 - g. Statistical Data blocks
 - h. Boolean blocks
 - i. Device Control blocks
 - j. Analog Alarm blocks
 - k. Digital Alarm blocks
 - l. Pareto blocks
 - m. Text blocks
 - n. Program blocks

NOTE: Analog Register and Digital Register blocks do not require SAC processing. Instead, iFIX processes them only when an operator opens a picture containing a link to either block or when a script that references either block runs. When either event happens, iFIX processes the blocks before the Ramp block.

3. When SAC scans a primary block, it:
 - a. Processes that block's entire chain.
 - b. Scans the next primary block.
4. SAC scans primary blocks of the same type, scan time, and phase in the order these blocks occur in the database, as the following example shows.

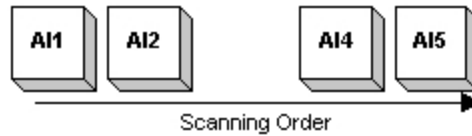
Example: Understanding a Database's Scanning Order

Suppose you create the five identical Analog Input blocks, shown in the following figure. SAC scans these blocks in the order shown.



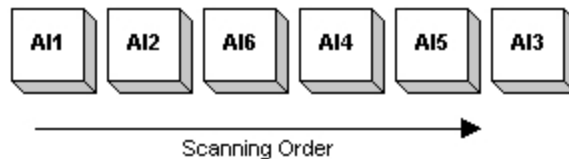
Blocks Assigned to a Scan Sequence

Now, assume you delete the third Analog Input block because you no longer need it. This creates an empty space in the database, as the following figure shows. SAC now scans the blocks as shown.



Blocks Deleted from a Scan Sequence

You subsequently discover you need to add two new Analog Input blocks, AI6 and AI3. The first new block, AI6, fills in the space left by deleting a block. The second new block, AI3, is added after AI5. SAC now scans the blocks in the order shown by the following figure.



Blocks Added to a Scan Sequence

Changing the Order of Solve

You can change the order of solve by exporting the database and editing the export file. Database Manager exports primary blocks in the order SAC scans all primary blocks. This results in all Analog Input blocks appearing first in the export file, followed by all Analog Output blocks, and so on. Once the primary block types are exported, Database Manager exports the secondary blocks.

For example, using the blocks in the [Sample Calculation Chain](#) figure, you can ensure SAC scans AI1 after AI2 and AI3 as follows:

1. Export the database.
2. Locate AI2's and AI3's entries in the export file.
3. Move both entries to the start of the file.
4. Save the export file and import it into an empty database.

For more information about completing these tasks, refer to the sections [Exporting Databases](#) and [Importing Databases](#).

Customizing the Toolbar

As an option, Database Manager lets you customize its toolbar by arranging and removing the toolbar buttons. Customizing the toolbar lets you display the buttons to frequently-used functions. For example, while developing a database you might display the buttons that let you add, modify, and delete blocks. However, when the database is complete, you might want to remove these buttons from the toolbar.

Once you remove a button, you can always add it back to the toolbar. Adding custom buttons is not supported. By default, the toolbar comes pre-configured with all the available buttons added for you.

NOTE: This feature is not available in Ribbon view.

Customizing the Tools Menu

Depending on your needs, you may want to start other applications after launching Database Manager. For example, if you routinely drag blocks to an Excel spreadsheet for modification, you may want to start Excel after Database Manager opens.

By customizing Database Manager's Tools menu, you can add menu items that launch any iFIX or third-party application. You can also arrange and delete these menu items as needed.

Adding a menu item requires you to enter:

- An application's path and name.
- The text that appears in the Tools menu.
- The arguments (command line parameters) you want to use when the application runs. As an option, you can configure Database Manager to prompt you for arguments when the specified application runs.

Once you specify this information, Database Manager creates the menu item and saves it as part of its default configuration. You can subsequently select the menu item to launch the associated application.

Index

A

- adding
 - blocks to the process database 53
 - multiple blocks to the process database 54
- alarm area database 7
 - setting up on each server 6
- alarms 5
 - described 4
- allocated blocks 73
- archiving 5
- Auto Refresh option 62
 - database spreadsheet 61
- automatic mode 36
 - described 35
 - function by block type 35

B

- batch blocks 25
- block fields 28
 - block name 28
 - described 28
 - formats 28
 - locating 29
 - naming convention 28
- block modes 35
 - described 35
 - function by block type 35

block types 75

abbreviations 76

blocks

adding 53

adding multiple 54

adding to Database Manager 3

allocating memory 74

batch 25

configuring 3

control 25

copying 8

copying and pasting 56

Database Dynamos 26

deleting 59

described 3

displaying information 58

duplicating 56

exception-based processing 31

function 3

how processed by SAC 4

importing 8

in the database spreadsheet 7

matching process steps to block types 15

modifying 58

moving 57

off scan 38

on scan 38

one shot processing 33

optional 22

pasting into Excel 56

- pattern type syntax 54
- phasing 33
- planning your automation strategy 21
- primary 22
- reaction to on/off scan changes 38
- saving 59
- scan order 78
- scan times 29
- scan times for large databases 20
- secondary 22
- sort order in database spreadsheet 70
- SQL 26
- Statistical Process Control (SPC) 25
- summary 26
- time-based processing 30
- types and descriptions 22
- boolean operators in queries 67
- building large databases 20

C

- chains 14
 - adjusting the design 16
 - described 3
 - designing 12
 - drawing a flowchart 16
 - how processed by SAC 4
 - maximum size 3
 - scanning order 78
 - understanding the design 20
 - writing an automation algorithm 14
- closing the process database 53

- collecting process information 10
- color 62
 - changing in a database spreadsheet 62
- columns 63
 - adding in a spreadsheet 63
 - creating a non-scrolling 62
 - modifying in a spreadsheet 63
 - removing from a spreadsheet 63
- configuration errors 71
 - checking the database for 71
 - resolving 72
- configuring blocks 3
- control blocks 25
- copying and pasting blocks in the process database 1
- creating 9
 - process database 9
- CSV 74
 - block data format conversion 8
 - Editing an export file 75
 - exporting process database to 74
- customizing 62
 - Database Manager toolbars 80
 - Database Manager Tools menu 81
 - database spreadsheet 62

D

- database
 - changing the scanning order 78
 - designing 11
 - displaying a summary 73
 - editing an export file 75

exporting 74

importing 77

order of solve 80

reloading 72

reloading from a Visual Batch script 73

verifying 71

Database Dynamos 26

function 26

toolkit 26

Database Manager 14

adding blocks 3

adjusting the chain design 16

block types 22

building large databases 20

configuring blocks 3

customizing a spreadsheet 62

customizing the Tools menu 81

described 1

designing a chain 12

designing an automation strategy 14

displaying a database summary 73

drawing a flowchart for the chain 16

exiting 7

gathering process information 10

matching process steps to block types 15

pausing a spreadsheet 61

queries 65

replacing data in a spreadsheet 61

right mouse menu 8

saving databases from a script or Program block 59

- saving large databases 21
- saving spreadsheet layouts 64
- searching a spreadsheet 60
- setting preferences 9
- sort orders 70
- starting 7
- toolbar 8
- understanding chain designs 20
- updating a spreadsheet 61
- using Go To in a spreadsheet 61
- working with analog and digital blocks 21
- writing an automation algorithm 14

database spreadsheet 7

- coloring 62
- commands 8
- creating a non-scrolling column 62
- customizing 62
- defining the layout 63
- editing 8
- loading layouts 64
- overriding the default layouts 64
- pausing 61
- properties 7
- replacing data 61
- restoring the default layouts 64
- saving layouts 64
- searching 60
- updating 61
- using Go To 61
- using in Database Manager 7

DBExporter.exe command 75

deleting blocks from the process database 58

designing 11

- chains 12

- process database 11

displaying

- any row in a spreadsheet 61

- block information from the process database 58

- blocks in a database spreadsheet 61

- columns in a database spreadsheet 61

- database summary 73

- rows in a database spreadsheet 61

duplicating blocks in the process database 56

E

editing 8

EGU limit 50

- precision and range 50

errors

- correcting verification 71

example 6

- Enviro company overview 6

Excel 56

- converting block data 8

- pasting blocks into 56

exception-based processing 31

- Analog Alarm block considerations 31

- Boolean block considerations 31

- Calculation block considerations 31

- Dead Time block considerations 31

- Digital Alarm block considerations 31

- Event Action block considerations 31
- Lead Lag considerations 31
- PID considerations 31
- Program block considerations 31
- Signal Select block considerations 31
- Statistical Data block considerations 31
- understanding 31

export file

- editing 75
- exporting process data to 74

exporting a process database 74

F

field formats 29

freezing columns in a spreadsheet 62

G

GDB 74

- editing an export file 75
- exporting process database to 74

Generate Wizard 54

- adding blocks 53
- adding multiple blocks 54
- automatic block name assignment 54
- customizing block fields 54
- entering patterns 54

Go To 61

- using in a spreadsheet 61

H

historical trends 5

- function 5

I

I/O data 4

- and the database 3

- how processed by SAC 4

I/O drivers

- defining an OPC server 49

- described 39

- OPC Client driver 41

- signal conditioning 49

- SIM driver 41

- SM2 driver 44

iFIX 7

- installing 6

importing a process database 77

importing blocks 8

L

layout 63

- defining in a database spreadsheet 63

loadable blocks (see Database Dynamos) 26

loading 64

- queries 69

- spreadsheet layouts 64

locking columns in a spreadsheet 62

M

manual mode 36

- described 35

- function by block type 35

memory 21

- allocating for blocks 73

- building large databases 20
- saving databases from a script or Program block 59
- saving large databases 21
- modifying blocks in the process database 58
- moving blocks to the process database 57

N

- non-scrolling column 62
 - creating in a spreadsheet 62

O

- objects 26
 - Database Dynamo 26
- off scan blocks 38
- on scan blocks 38
- one shot processing 33
- OPC Client driver 41
- OPC server 49
- opening the process database 53
- optional blocks 22
 - list 22
- order of solve 80
- overphasing blocks 35
- overriding 64
 - default query 69
 - default spreadsheet layout 64

P

- pasting blocks into Excel 56
- pattern types 55
 - syntax 55
- pausing a spreadsheet 61

PAUT mode 37

function 37

understanding 37

phasing 33

assigning 34

second and subsecond blocks 34

what it does 33

planning 10

block scan times 20

designing the process database 10

gathering process information 10

PMAN mode 37

function 37

precision

EGU limits 50

preferences 9

Database Manager 9

primary blocks

described 22

function 23

scan times 29

standard 23

summary 26

process analyzing 13

process data

archiving 5

trending 5

process database

adding blocks 53

adding multiple blocks 54

adjusting the chain design 16

basic operations 52

before you begin creating 6

building large databases 20

changing the scanning order 78

closing 53

copying and pasting blocks 56

creating 9

customizing a spreadsheet 62

deleting blocks 59

described 1

describing 12

designing 11

designing a chain 12

designing an automation strategy 14

displaying a summary 73

displaying block information 58

drawing a flowchart for the chain 16

duplicating blocks 56

editing an export file 75

example 6

exporting 74

functions 2

gathering process information 10

importing 77

matching process steps to block types 15

modifying blocks 58

moving blocks 57

opening 53

order of solve 80

- overriding the default spreadsheet layout 64
- pausing a spreadsheet 61
- reloading 72
- reloading from a Visual Batch script 73
- replacing data in a spreadsheet 61
- restoring the default spreadsheet layout 64
- sample application 11
- saving blocks 59
- saving databases from a script or Program block 59
- saving large databases 21
- saving spreadsheet layouts 64
- searching a spreadsheet 60
- understanding chain designs 20
- updating a spreadsheet 61
- using 5
- using Go to in a spreadsheet 61
- using the database spreadsheet 7
- verifying 71
- working with analog and digital blocks 21
- writing an automation algorithm 14

processing large databases 20

Q

queries 67

- appending 68
- boolean operators 67
- described 65
- editing 68
- expanding 69
- grouping 68
- loading 69

- overriding the default 69
- refining 69
- relational operator examples 67
- relational operators 66
- restoring the preset default 69
- saving 69
- syntax 66
- wildcard examples 67
- wildcards 67

R

- range
 - EGU limits 50
- refreshing the screen 62
 - database spreadsheet 61
- relational database 5
 - functions 5
- reloading 72
 - database from a Visual Basic script 73
 - process database 72
- replacing data in a database spreadsheet 61
- restoring 69
 - default spreadsheet layout 64
 - preset default query 69
- right mouse menu 8

S

- SAC 30
 - block phasing 33
 - changing the database scanning order 78
 - exception-based processing 31

function 4

one shot processing 33

processing I/O data 4

time-based processing 30

saving 64

blocks to the process database 59

databases from a script or Program block 59

queries 69

spreadsheet layouts 64

SCADA servers 5

establishing a connection from Database Manager 53

loading process database from 72

monitoring and reporting functions 4

trending and display capabilities 5

scan time

block reaction to on/off scan changes 38

described 29

examples 30

exception-based 31

one shot 33

time-based 30

scanning order

changing in the database 78

described 79

Scheduler 5

adjusting database values using 5

functions 5

SCU 7

configuring each server 6

searching a database spreadsheet 60

secondary blocks

- described 22

- function 24

- standard 24

- summary 26

sharing blocks among process databases 56

signal conditioning 49

SIM driver 41

- analog values 41

- described 41

- digital values 41

- generating random values 43

- signal generation registers 43

- using 41

slot numbers 28

SM2 driver 45

- C API 45

- generating bad data 45

sort order 70

- adding a column 70

- arranging columns 70

- changing the default 71

- described 70

- loading 70

- removing a column 70

- saving 70

sorting

- columns in database spreadsheet 70

- disabling in database spreadsheet 70

SQL blocks 26

standard blocks 26

summary 26

Statistical Process Control (SPC) blocks 25

syntax 66

database query 66

T

time-based processing 30

assigning a scan time 30

examples 30

understanding 30

toolbar 8

customizing Database Manager 80

Database Manager 8

Toolkit 26

Database Dynamo 26

Tools menu 81

customizing in Database Manager 81

trending 5

historical values 5

U

unfreezing columns in a spreadsheet 62

unlocking columns in a spreadsheet 62

updating a spreadsheet 61

used blocks 73

using the process database 5

V

VBA 73

adjusting database values using 5

functions 5

reloading a database from a script 73

verification messages 72

list 72

resolving 72

verifying a process database 71

W

Workspace 5

reloading a database from 73

using to provide visual cues for alarms 4

writing an automation algorithm 14