

Previs Inc.

Users Manual

Bailey DCS Simulator

August 2011

Prepared by:



Previser Inc.®
Managing Critical Microsoft® Technology

Printed in Canada

This document is subject to continuous improvement, and as such is subject to change without notice. Feedback or inquiries regarding this document are welcome. Contact Previser Inc. via www.previser.com.

Proprietary Notice: This document and related software contain proprietary information which represents trade secrets of Previser Inc.® and may not be copied or disclosed, except as provided in the license with Previser Inc. Use of the information in this document and related software, for the reverse engineering of OPsCon,™ or for the development or manufacture of similar software is prohibited. The information in this document is subject to change without notice and should not be construed as a commitment by Previser Inc. Previser Inc. assumes no responsibility for any errors that may be in this document.

Copyright © 2002 - 2011 Previser Inc.® All rights reserved.

Unauthorized reproduction is a violation of Previser Inc. copyright.

Trademarks

Previser® and OPsCon™ are trademarks or registered trademarks of Previser Inc.®

Bailey,® Network 90,® Net 90®, Batch 90® and Infi 90® are registered trademarks of ABB

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

Notice

Previser Inc., its partners, affiliates, employees, and agents, and the authors of, and contributors to, this publication and the software it represents, specifically disclaim all liabilities and warranties, express and implied (including warranties of merchantability and fitness for a particular purpose), for the accuracy, currency, completeness, and/or reliability of the information contained herein, and/or for the fitness for any particular use, and/or for the performance of any material, and/or for equipment selected in whole or part by the user in reliance upon information contained herein. Selection of materials and/or equipment is at the sole risk of the user of this publication.

Table of Contents

Bailey DCS Simulator Users Manual	1
1 Introduction	1
2 Functions and Features	2
3 Install, Configure and Operate	7
4 The BaileySimClient.....	15
Appendices	28
Appendix A – Function Code Support	28
Appendix B – Configuring the <i>Default.INI</i> File	35
Appendix C – Event Log and Message Window Messages.....	49
Appendix D – Process Simulator Interface (API)	53
Appendix E – Support for Specific Connected Products.....	54
Appendix F – Auto-Create CFG File	55
Appendix G – Details of Loop Back Mode	57
Appendix H – MCP02/ICT03 DIP switch settings.....	61
Appendix I – Format of Points Table Report.....	63
Appendix J – Setup RTTgSCSI.ini for SCSI channel.....	65
Appendix K – Set Up DCSIOManager	66

Bailey DCS Simulator Users Manual

1 Introduction

Key Characteristics

Key characteristics of this simulator are:

- Execute Bailey DCS controller module CFG files *as-is*
- Simulate to 400+ controller modules simultaneously
- Support MFP/BRC controller types and older ones as well
- Supports 150+ of the most commonly used Function Codes¹
- Supports emulated CIU (various types) via serial or SCSI
- Supports connection of ABB and third party operator consoles²
- Supports connection of ABB and other engineering tools³
- Load and unload controller module programs (CFG) with ease
- Supports ABB Batch 90 program emulation.
- Executes as service within Microsoft Windows environment

Operator Training Simulator Applications

- Easily connected to plant process simulation software
- Provides API for IO simulation
- Supports freeze, resume, save/restore process state
- Supports command logging and replay
- Supports alarm management functions
- Throttle from 0.1 X real time to 10 X real time speeds⁴

HMI Test/FAT Applications

- Checkout and verification of consoles prior to installation
- Provides specific test functions for HMI test.

Other Applications

Give your control development engineers a personal use desktop facility, connected directly with EWS Tools, to directly execute control programs with optional connection to a simulated plant process. All without the cost of putting all the DCS hardware in place

¹ Additional function code support is available.

² Refer to Appendix for list of known operator console support.

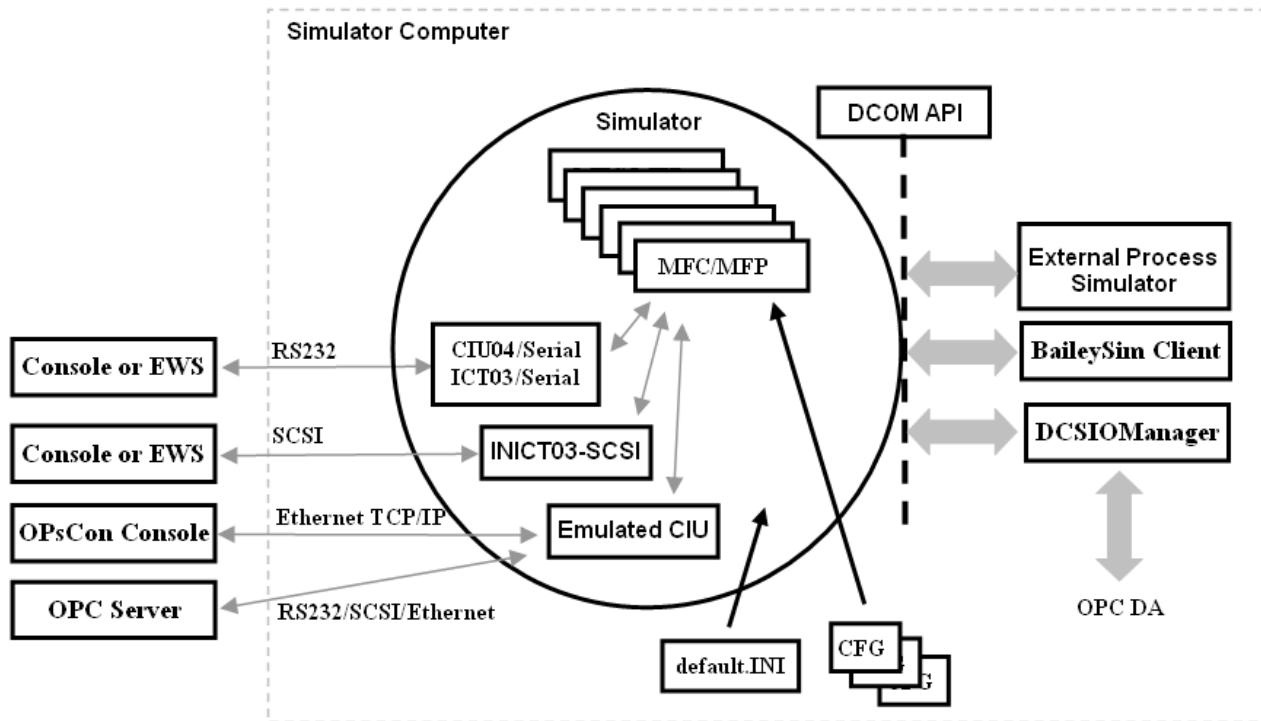
³ Refer to Appendix for list of known engineering tools support.

⁴ Large systems may be platform or performance limited. Contact Previsé for performance assessment.

2 Functions and Features

General

The following diagram illustrates the Bailey DCS Simulator.



In this diagram, the key features are:

- **default.INI** file used to configure Bailey DCS Simulator
- Controller **CFG** files loaded to multiple simulated controller modules (MFP/BRC & others)
- Console/EWS/OPC server devices connected by emulated CIU, with support for RS-232 (CIU04 and INICT03), SCSI (INICT03) and Ethernet (to Previs OPC server and console)
- External process simulator connected via special purpose API (refer to technical manual for API)
- **BaileySim** Client connects via API and provides a general purpose simulator user interface
- **DCSIOManager** provides wraparound IO via API, and provides both OPC Data Access v2.0 client and server interface.

Configuring the DCS Structure

Configure the DCS structure via an INI file. This INI file defines what Loop : PCU : Module structure the simulated Bailey DCS will have. This INI file defines what CIU devices and MFP/MFC controllers are simulated. Refer to the Appendix for further details of this INI file.

Operates as a Service

The Bailey DCS Simulator executes as a service under the Microsoft operating system (W2000, W2003 or XP) and may be configured for:

- Automated startup at Microsoft system boot.
- Manual startup on demand

Regardless of how it is started, the Bailey reads the *Default.INI* file at startup, and automatically configures its internal DCS structure, automatically loads the controller program (CFG) files, and automatically starts operation (i.e. simulation of Bailey DCS).

Computer Interface Unit (CIU) Devices Supported

The Bailey DCS Simulator provides an emulated CIU (Computer Interface Unit) and supports up to eight simultaneous emulated CIU devices of different types, depending on the license options you have purchased.

You may include 0, 1 or 2 emulated INICT03-SCSI interfaces, with a high speed SCSI connection. The remaining CIU can be any combination of:

- Emulated NCIU04/INICT01/INICT03 via RS-232 serial COM port
- Emulated TCP/IP CIU for the Previs OPCon operators console or OPC server, or for other variations of OPCon, executing either on the same computer or a different networked computer.

Connected software and devices will behave almost as if connected to a real CIU, including startup behavior, normal operator functions, alarms, trends, controls, block tuning, specifications, exceptions, and other aspects.

NOTE: *Though a real CIU is limited to 19.2 Kbaud, the emulated RS-232 CIU within the Bailey DCS Simulator can run at higher speeds. The Bailey DCS Simulator supports all baud rates supported by the underlying PC hardware platform.*

Refer to the Appendix for a complete list of the software and devices which are known to work with the simulated CIU connections.

Controller Devices Supported

The Simulator simulates one or more controller modules, with all logic functions running as they would in the real DCS, at real time or faster than real time. The Simulator supports a wide range of controller types (see Appendix B Table B-1). Note however that Function Code differences between controller types is not supported. A Function Code will execute the same regardless of controller type in which it is used.

System Size

The largest configuration executed to date has 95 controller modules with 170,000+ blocks. Performance is normally limited only by computer CPU speed, memory type, and operating system. Contact Previsé if you have high performance requirements above 50 controllers or 200,000 blocks of logic. Previsé expects that applications to 300,000+ blocks can be handled.

Programming the Controllers

Program the simulated DCS controllers by loading the binary CFG controller program files automatically at Simulator startup. This is accomplished by configuring the *Default.INI* file to provide the file path to each controller CFG file.

You may also program the controllers via the CIU using WinTools, CADEWS or ComposerIT Engineering Workstation (or other) to load the CFG file the same as you would with a real DCS.

NOTE: Unless *OPTIONS* switch in *default.INI* file is set to save CFG files any changes made to the CFG files is lost at exit from simulator.

Block Tuning Functions

The simulator supports block tuning to change tunable specifications of controller function codes both from console and EWS software.

Exception Handling

The Simulator emulates the function of each function block in the controller modules as accurately as possible. This includes the generation of exception packets when data changes within any of the exception generating block types. This includes data exceptions, alarm state changes, and specification exceptions.

The Simulator emulates the Bailey DCS *Loop:PCU:Module* structure. The exception packets get passed from the originating module to all destination modules the as they do in the real Bailey DCS.

Exception packets are handled by CIU devices in the same manner as with the real DCS, so that an operators console connected to the simulated CIU will receive exceptions much like in a real DCS.

Loop Back Mode

The Bailey DCS Simulator supports a Loop Back Mode that is designed specifically to provide semi-realistic operator console interaction, but without the requirements for any simulated process.

Without this mode, the process state within the Bailey DCS Simulator would normally prevent dynamic data from being provided to the console in the absence of a full process simulation, which is impractical for most console testing. However, the Loop Back Mode provides dynamic data and interaction for all console block types.

Refer to the Appendix for details of the Loop Back mode operation.

HMI Test Functions

The Bailey DCS Simulator includes a function (i.e. Test tab) specifically designed to help test new HMI systems connected to the Bailey DCS. This will assist to commission HMI systems by helping verify that each screen element is connected to the correct block address.

Auto Generate CFG Files for Bailey DCS Simulator

If you are using the Previs OPC Server for Bailey DCS, the Bailey DCS Simulator provides a tool that lets you automatically create a set of CFG files to populate the Bailey DCS Simulator with control logic that matches your console configuration. These CFG files are created from the OPC Server CSV tag database file. You would normally use these CFG files with Loopback Mode enabled.

Alarm Enable/Disable

The BaileySim Client is provided with the ability to enable or disable alarms for each tag type, and to force each individual supported alarm type. This is provided to provide the means to test the alarm system on connected HMI systems.

Watch List

The BaileySimClient Watch List function lets you monitor a set of user defined block addresses. The values in the list update automatically. Save and Restore the Watch List

Force List

The BaileySim Client Force List function lets you force a set of user defined block addresses to specific values, irrespective of the values in upstream logic. Force and Unforce values at will. Save and Restore the Force List.

Alarm Management

The Bailey DCS Simulator supports several alarm management features (Refer to ALARMS line in default.INI file). These features are intended to help manage the alarms within consoles attached to the Bailey DCS simulator. The following specific features are supported:

1. To mitigate initial flood of alarms at startup, all alarms can be disabled at simulator startup, and enabled at a later time.
2. Alarm ACK is shared from console to console so that alarm does not need to be acknowledged more than once (currently supported for selected consoles on SuperLoop only).
3. A global alarm ACK function supports acknowledgement of all alarms on all consoles on a manual or automated basis.

Batch 90 Program Emulation

Batch 90 program emulation is supported and a Batch Debugger is provided to assist in debug of Batch 90 programs.

The USB License Key

The USB license key is used to configure the simulator for the license limits that you have purchased. Without the USB license key, the simulator will not operate.

The types of limits imposed by your license key include:

- Total number of modules.
- Total number of function blocks in the DCS.
- Total number of CIU devices
- Module Mode control enabled/disabled
- Enable or disable SCSI communications (simulated INICT03)
- Enable or disable API functions.
- License expiry date (may be either a specific date or unlimited)

For example your license may only allow you to run a maximum of 3 modules and a total of 5,000 function blocks. It can also limit the ability to switch between execute and configure modes.

You can check what kind of a license you have by opening the BaileySimClient and switching to the Messages tab. You will see messages describing all license restrictions.

3 Install, Configure and Operate

What You Should Have Received

When you purchase the Bailey DCS Simulator, you should receive:

- A CD-ROM containing the software and documentation
- A USB License Key to unlock the software according to your purchased license.

If you intend to install a SCSI connection from the Bailey DCS Simulator to an operator console or engineering workstation, then you should also have purchased and receive:

- SCSI adapter card to install with Bailey DCS Simulator. This is *not* a standard SCSI card and *must* be purchased from Previs.
- SCSI adapter card to install with operator console or engineering workstation software. Purchase this from the vendor for the console or EWS software or from Previs.
- SCSI cable to connect from one SCSI card to the other.

Minimum System Requirements

The Bailey DCS Simulator is a CPU intensive application and the host computer must meet a high performance standard.

- Intel Pentium 4, 3.0 GHz minimum, 2MB L2 Cache
- 4GB, 500Mhz SDRAM is recommended
- USB port
- Microsoft Windows XP SP2 (Hyper threading ENABLED) or Windows 2003.
- NTFS file system configured
- Two (2) separate hard drives
 1. Minimum 30 GByte
 2. Minimum 80 GByte
- 48X CD-ROM
- ONE USB port minimum (for license key)
- 56K v.92 Data/Fax V92DF
- 100BaseT networking

NOTE: Contact Previs to confirm the computer platform if your application needs (a) more than 50 controller modules, or (b) more than 100,000 function blocks, or (c) SCSI CIU emulation.

For applications greater than 25 controllers or 50,000 function blocks, the computer that hosts the Bailey DCS Simulator must be dedicated to that purpose, and should not have other applications installed.

For further details refer to the Technical Specification, Bailey DCS Simulator Host Platform on the CD-ROM.

Installation and Configuration

Step 1 – Install the SCSI Adapter Cards If Applicable

If your application includes a SCSI emulated CIU, then power down the host platform, open in up, and install the (one or two) SCSI adapter cards provided. Close up the computer again and install the software.

Step 2 – Install the Bailey DCS Simulator CD-ROM

To install and configure the Bailey DCS Simulator CD-ROM:

1. Do not install the USB License Key yet.
2. If installing an update to a previous Bailey DCS Simulator installation, stop the Bailey DCS Simulator first.
3. Run the Setup.exe executable from the main directory on the CD. The only configurable option is the target installation directory.
4. Verify that the Bailey DCS Simulator appears in the list of installed services accessed via the Control Panel.

Step 3 – Install the USB License key

NOTE: The demo version of the Bailey DCS Simulator, does not require a USB License key. You can skip this installation step.

However, if you have purchased the simulator and received a USB license key you must complete this installation step to ensure that your computer recognizes the USB License Key.

To install and configure the USB License Key:

1. DO NOT install or plug in the USB key yet !
2. Run CBSetup.exe from the directory where Bailey DCS Simulator was installed.
3. At dialog box *“Would you like to install CRYPTO-BOX support?”* select “Install” and press the “OK” button.
4. At dialog *“This setup will install CRYPTO-BOX support on your computer. Would you like to proceed?”* press “YES”.
5. At dialog *“Select MARX Software”* select “CRYPTO-BOX USB CryptToken(USB)” option and press “OK” button.
6. Press “Ok” button.

7. Install the USB License Key into your Computer. At this point the “Hardware Wizard” should pop-up to give you a chance to complete the USB installation.
8. On “*Hardware Wizard*” dialog box press “Next” button
9. Press “Continue Anyway” button and then press “Finish”.
10. At this point the USB License Key install is complete.
11. Verify that the process has completed properly by rebooting your computer with your license key installed, and then by starting the Bailey DCS simulator. Review the Event Log or the BaileySimClient Debug Screen messages to see the size of the DCS you are permitted (number of modules, number of blocks and number of CIU). These should match with the limits you expect on your license key.

Step 4 – Run HwInstall.EXE for SCSI Installations

NOTE – This step is for installations with SCSI emulated CIU only.

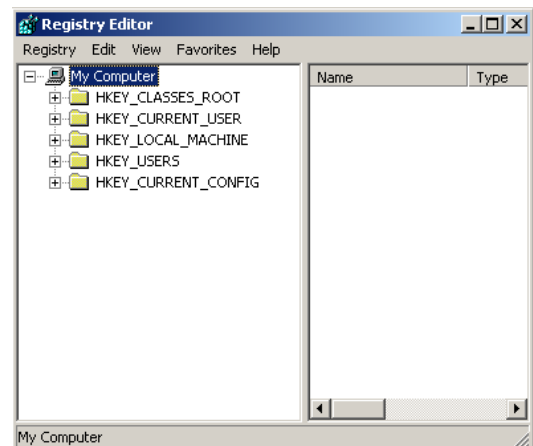
1. Locate the directory `.. \ HwInstall \` on the CD-ROM. Copy this directory into the `.. \ BaileyDCSSimulator \..` directory as installed in Step 2 above.
2. Locate the file `hwinstall.exe` on the CD-ROM.
3. Double click this file to execute it.

Step 5 – Registry Verification

NOTE – This step is for installations with SCSI emulated CIU only.

Verify that the registry has been correctly updated as follows:

1. Start the registry edit application called *RegEdit* by typing `regedit` at the Windows Start | Run dialogue.
2. Locate the registry key:
 - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\VscsiPCI (VscsiP2K, VscsiU3W or VscsiU3K)
 - Change the “start” key-value from “1” (System device) to “0” (Boot device) (Unless it’s already set correctly)



3. Locate the registry key:
 - HKEY_LOCAL_MACHINE\SYSTEM\Current Control Set\Services \sym**
 - There may be multiple of these at the same path. Verify each of them.
 - Change the “start” key-value to “4” (Unless it’s already set correctly)

Step 6 – Power down and Restart computer

Shut down and power the computer all the way off. Then restart it.

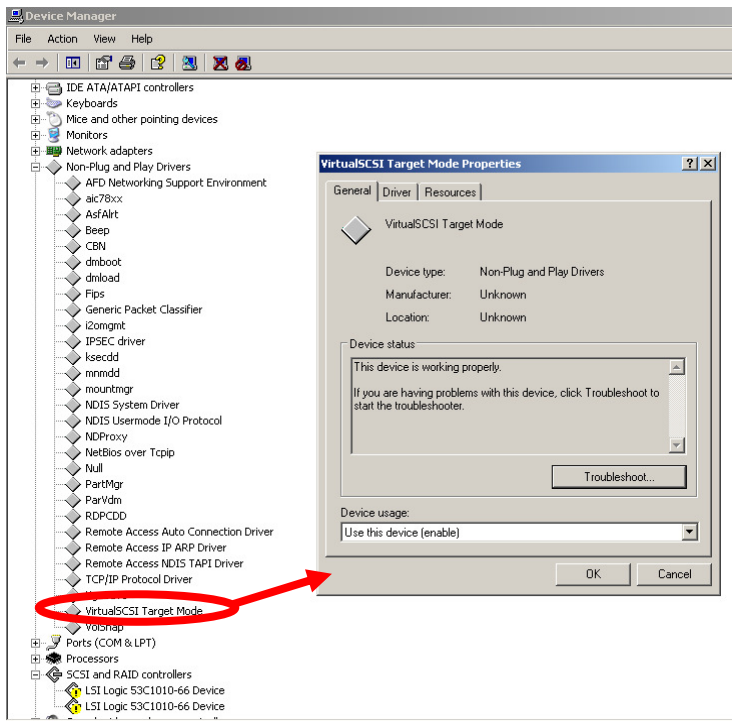
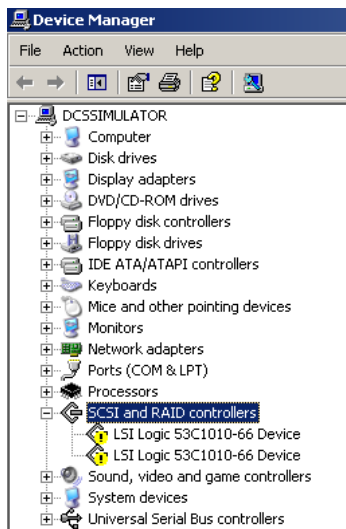
NOTE – For installations with SCSI emulated CIU, be sure to power the computer all the way off (i.e. remove electrical power).

Step 7 – Verify Device Manager Settings

NOTE – This step is for installations with SCSI emulated CIU only.

Verify that the SCSI device is installed and ready to go as follows:

1. Start the Microsoft Device Manager (procedure varies by OS version)
2. Verify that you have one or two LSI Logic 53C1010-66 Device entries at SCSI and RAID controllers. Do not worry about yellow highlight on these devices.
3. Set Device Manager to show hidden devices.



4. Verify that the Plug and Play driver for **VirtualSCSI Target Mode** is present as shown. Double click and verify that it is working normally as shown.

NOTE: Always power down and remove electrical power from the Bailey DCS Simulator computer before connecting SCSI cables.

Step 8 – Configure the DCS Via the DEFAULT.INI file

Configure the Bailey DCS Simulator via the **default.INI** file in the **.. |Program Files |Bailey DCS Simulator |..** directory. Locate this file and edit the file to configure the Simulator for your specific DCS configuration. Refer to the Appendix for details of how to do this.]

Step 9 – Configure (Optional) Batch 90 Emulation

If you wish to execute Batch 90 programs within the Bailey DCS Simulator then:

1. Your license must support Batch 90 Emulation.
2. You must set the OPTIONS switch in default.INI to ENABLE Batch 90 support
3. Refer to the manual entitled **Users Manual, Bailey DCS Simulator, Batch Addendum** for further details about use and operation of the Bailey DCS Simulator Batch 90 functions.

NOTE: *To access the Batch 90 functions within the Bailey DCS Simulator, your license must include Batch 90 Support.*

Installing an Update

If you already have the Simulator installed, and you receive an update, you must follow this procedure to install the updated version:

1. You should not need to install, or reinstall, support for the USB key if this has been completed once before
2. Shutdown and power down all computers that are connected to the Simulator Computer. The purpose of this is to ensure that no application continues to connect to the Bailey DCS Simulator application from any other computer.
3. Close all applications that are running on the Simulator Computer.
4. On the Simulator Computer, locate the Bailey DCS Simulator in the list of services installed (via Control Panel > Services list). Once you locate the Bailey DCS Simulator then STOP the Simulator and set the Simulator to be started MANUALLY at system boot.
5. Locate and backup the **default.INI** file in the **.. |Program Files |Bailey DCS Simulator |..** directory. This file, which

contains your DCS configuration, will be lost at installation of a new version.

6. Uninstall the simulator using Control Panel Add/Remove Programs function. There is no need to select deletion of all files
7. Insert new Bailey DCS Simulator CD-ROM and locate and execute the SETUP.EXE file. The only installation option is the directory to install the files, which should be at the same location as the previous version.
8. Restore the **default.INI** file that you previously backed up to the **..\Program Files\Bailey DCS Simulator\..** directory.
9. The new simulator version is now installed are ready for use.
10. On the Simulator Computer, locate the Bailey DCS Simulator in the list of services installed (via Control Panel > Services list). Once you locate the Bailey DCS Simulator then START the Simulator (if you wish) and set the Simulator to be started AUTOMATICALLY at system boot (if you wish).

Start The Simulator

NOTE: You must ALWAYS start the Bailey DCS Simulator BEFORE starting any EWS or operator console software connected to it.

If you are using a SCSI CIU is particularly important to power the Bailey DCS Simulator AND connected console or EWS products all the way to clear any hardware register states before starting up.

If the Simulator is configured for manual startup (reference Service settings), then the procedure to start the Simulator is as follows:

1. From the Control Panel locate the Services list.
2. On the Services List locate the Bailey DCS Simulator, and right click on the item.
3. Select START.
4. The Bailey DCS Simulator will now start.

If the Simulator is configured for automated startup (reference Service settings), then the procedure to start the Simulator is as follows:

1. Reboot the Microsoft operating system.
2. The Bailey DCS Simulator will auto-start at system boot.

Regardless of which method you use to start the Simulator, the simulator will automatically load the DCS configuration as specified by the **Default.INI** file at startup.

Verify that Simulator is Running Normally

to verify that the Simulator is running normally, you should check the following:

1. From the Start menu, start the BaileySimClient and look at the Message Tab. At startup, the Message tab should include the following lines:
 - *File XX (T: <module type><L>:<P>:<M>* for each CFG file loaded. You must verify that CFG files load without error.
2. At connection of an operator console to a simulated serial CIU, the content of the Message tab should include the following lines:
 - *Command Environment*
 - *Command Restart*
 - *Command Online/Offline*
3. At connection of an operator console to a simulated TCP/IP CIU, the content of the Message tab should include the following lines:
 - *New Telnet CIU address XX:XX:N*
 - *Connection Accepted*
 - *Command Environment*
 - *Command Restart*
 - *Command Online/Offline*
4. If a SCSI CIU is installed verify that the SCSI channel starts correctly. A sample (for OIS2 with VAX and VMS) is provided here. Refer to Appendix for a description of each record.
 - *RTg(SCSI1:4:0): ADLL: VScsiV2.dll version: 6, 2, 8, 0 detected!.*
 - *RTg(SCSI1:4:0): #1 - Emulation now running!.*
 - *RTg(SCSI1:4:0): RTTg SCSI1:4:0 Emulation running.*
 - *Inquiry CDB Received.*
 - *Inquiry CDB Received.*
 - *Inquiry CDB Received.*
 - *Inquiry CDB Received.*
 - *#2 - Emulation channel has been Reset!.*
 - *#1 - Emulation now running!.*
 - *Inquiry CDB Received.*
 - *Inquiry CDB Received.*
 - *Inquiry CDB Received.*
 - *Inquiry CDB Received.*
 - *Inquiry CDB Received.*
 - *CIU(5:2:2) Command Restart.*
 - *CIU(5:2:2) Command Restart.*
 - *CIU(5:2:2) Command Environment(F_1).*
 - *CIU(5:2:2) Command Restart.*
 - *CIU(5:2:2) Command OnlineOffline*

NOTE: The actual lines you see will vary somewhat depending on what console you are using and which SCSI port you are using.

5. Alternatively you may review the Microsoft Event Log. The appendix provides details of all Message tab and Microsoft Event Log entries.
6. You **MUST** start the Simulator before starting any console that may be connected to the Simulator. Wait a minimum of 10 seconds before connecting any connected console.

Shutdown

If the Simulator is configured for manual startup (reference Service settings), then the procedure to start the Simulator is as follows:

1. Stop consoles and EWS devices before stopping the Bailey DCS Simulator
2. From the Control Panel locate the Services list.
3. On the Services List locate the Bailey DCS Simulator, and right click on the item.
4. Select STOP.
5. The Bailey DCS Simulator will now stop.

If the Simulator is configured for automated startup (reference Service settings), then simply shutdown the computer to stop the Simulator.

Connecting an OPsCon Console via TCP/IP

To connect an OPsCon console via TCP/IP refer to instructions in the OPsCon Configurator Users Manual. Ensure that the Simulator is started before you try to connect.

The Simulator was designed to act like real hardware. This means that once the OPsCon driver is talking to the Simulator, the simulated functionality includes: Control functions, Exception Reports, Tuning functions, CPU functions, module mode control, time synchronization and more.

Connecting Devices Via Serial CIU

Once the Bailey DCS Simulator is started, the serial COM ports (or SCSI ports) you have selected to configure as a CIU, will automatically behave like the selected CIU type when a console or EWS connects to it.

You may need to configure your console or EWS (engineering workstation) product. If so, follow the same vendor instructions as if you were connecting your device to a real CIU.

Ensure that the baud rate and COM port settings match for both devices connected, and ensure that a null modem cable is used for computer to computer connections.

4 The BaileySimClient

The BaileySimClient provides a graphical interface to the Bailey DCS Simulator.

To run the BaileySimClient application:

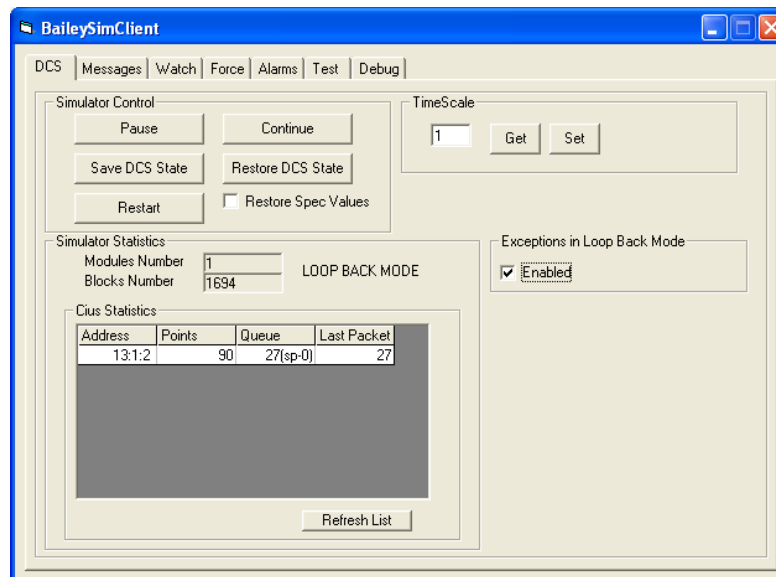
1. First verify that the Bailey DCS Simulator is already started and running as a service,
2. Start the BaileySimClient from the Start menu.

General

The BaileySimClient has the following tabs:

- DCS Tab
- Messages Tab
- Watch Tab
- Force Tab
- Alarms Tab
- Test Tab (Loopback mode only)
- Debug Tab

DCS Tab



This tab gives you the ability to pause and resume the simulator, and provides the ability to save and restore a DCS state and provides general simulator statistics so that operating state may be observed, and provides further options.

Simulator Controls

The Simulator Controls on this tab include:

1. **Pause** – This will pause execution of simulator controller execution at the end of the current cycle. CIU functions will continue unchanged, but function code execution stops. DCS data remains unchanged.
2. **Continue** – This resumes execution of controller function.
3. **Save DCS State** – This opens a dialogue box to select a file name to save the current DCS state. The DCS must be in PAUSE state before attempting to Save the DCS State.
4. **Restore DCS State** – This opens a dialogue box to select a file name from which to restore the DCS state. The DCS must be in PAUSE state before attempting to Restore the DCS State. The current DCS configuration must match that when the state you wish to restore was saved.
5. **Restore Spec Values** – If not checked then Function Code specifications are taken from CFG files at time of restore. If checked then specifications are restored from saved DCS file. The default is to NOT restore specifications.
6. **Restart** - This performs a cold restart of simulated DCS, causing the initial state (saved automatically at simulator start), to be restored (e.g. as in Restore DCS State). As this function will cause all DCS state to be lost, you will be asked if you wish to continue before the function is committed. Console communications is not affected.

NOTE: The Restore Function will restore Function Code specification values if **Restore Spec Values** is selected. If specifications are restored they will overwrite any specifications that were loaded when the CFG files were loaded at simulator startup. This may be confusing. It is recommended that specifications NOT be restored in most cases.

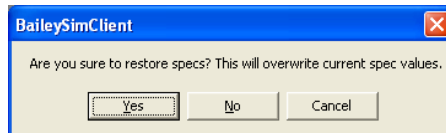
To Save a DCS State you must:

1. Click PAUSE to pause DCS operations.
2. Click Save DCS State to save the current DCS state. You will be prompted for a file name. It is recommended you choose names that describe the state being stored.
3. Click Continue to continue DCS operations.

To Restore a DCS State you must:

1. Click PAUSE to pause DCS operations.

2. Click Restore DCS State to restore the selected DCS state. You will be prompted for a file name. The DCS file must have been generated with the same DCS configuration you intend to restore to avoid error.
3. Click Continue to continue DCS operations.
4. If you select to restore spec's, you will receive a dialogue box to confirm that you wish to restore specifications.
5. Confirm that the new DCS state file restored correctly by reviewing the messages at the Message Tab.



Time Scale

This function supports a *speed throttle* for simulated DCS operations, in the range of **0.1 X** normal speed (i.e. normal real time speed) to **10 X** normal speed (i.e. 10 times faster than real time). The default is **1 X**.

This will have the effect of making the DCS “seem” to work faster (or slower) than the real DCS does. Use this as a means to decrease wait times during “timeout” or “soak” operations, or to speed certain training activities. Reduce the time it takes to generate the various training scenarios when starting your plant from a blank initial state.

CAUTION – This throttle function is sensitive to CPU speed and CPU utilization on your computer. When you first attempt to use this speed up function, increment the speed upwards slowly and watch the CPU usage in task Manager. **DO NOT** let the CPU utilization exceed about 70% as to do so may cause unexpected simulator results, and may require a computer reboot. If you have a computer with lots of spare CPU capacity this won't be a problem. However if your computer doesn't have enough capacity, you must avoid the higher settings.

Simulator Statistics

The Simulator Statistics include:

- The number of modules that have been successfully loaded.
- The total number of blocks that have been successfully loaded. Note that you'll need to review the event log to verify that there are no block errors.
- A list of all CIU's that have been instantiated, either via the *Default.INI* file or via TCP/IP connection. You will need to **Refresh** this list to see new CIU's after they are added. The list provides:

- The CIU **Address** as Loop:PCU:Module
- The number of **Points** (or tags) that have been established in this CIU. You will see this number increment throughout the initialization phase of CIU communications as a console connects. This number will likely remain 0 for WinTools and other engineering station connections.
- The number of data Exceptions currently waiting in the **Queue** for transmission to the computer connected to the CIU. If this number exceeds approximately 180 to 250 then you will observe some console delays. On systems with VERY large tag databases, you may see this number temporarily get quite large at startup, after a change in DCS state, or at the moment of background exception reporting.
- The number of data exceptions in the **Last Packet** sent to the console device. When there are a lot of data exceptions in the queue, this number will max in the range 180 to 250.

Exceptions in Loop Back Mode

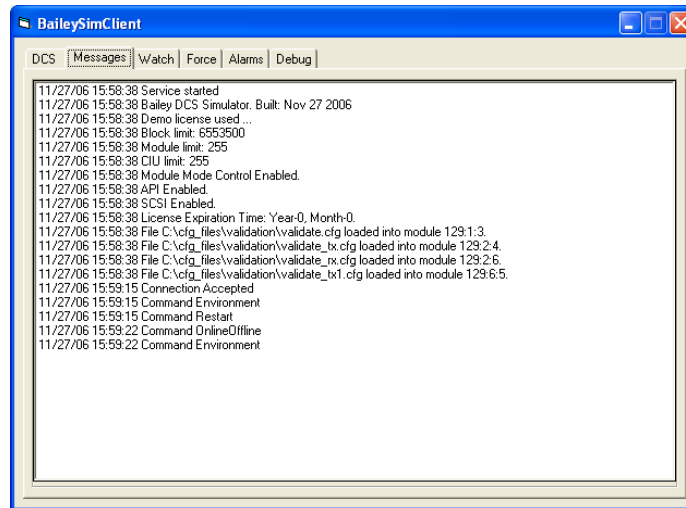
NOTE: This section of the screen is ONLY visible when **Loopback Mode** is selected via the **default.ini** file.

When the Bailey DCS Simulator is started in Loop Back Mode (refer to Default.INI file to see how):

- A Loopback Mode message is provided in the Debug Messages tab and in the Event Log to signify selection of this mode.
- Each console connected Function Code operates in a special Loopback mode. In this mode, most block inputs from within the DCS are ignored and simulated data is used. Refer to Appendix G on specific operations for each block type.

In Loopback mode all data for all console tags is constantly changing at a rate defined by default.ini file parameters. As a result, it is possible to configure the Bailey DCS Simulator to generate too many exceptions, such that the CIU connection cannot pass all of the exceptions generated. The precise point at which this happens is determined by the number of console tags, and by the cycle time parameters defined for the Loopback mode in default.ini. In Loopback Mode a check box is provided on the DCS tab to ENABLE/DISABLE the sending of data exceptions.

Messages Tab



Any messages that the Simulator generates will be displayed in here. This includes the license information like:

2:54:31:758 - "Valid license found."

2:54:31:758 - "Block limit: 300"

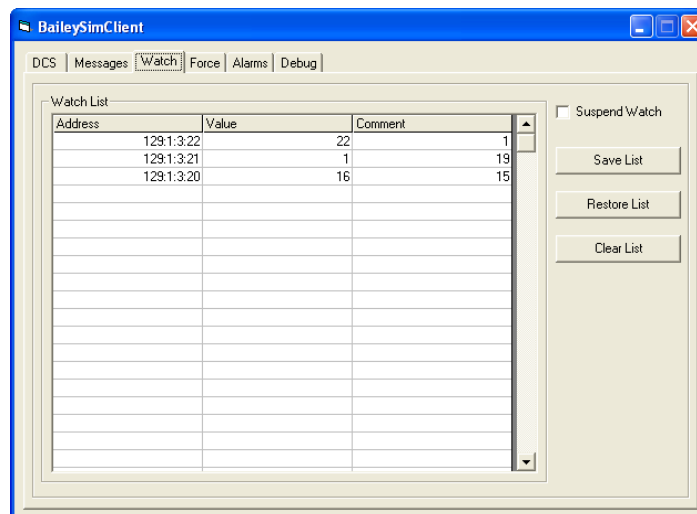
2:54:31:758 - "Module limit: 4"

2:54:31:758 - "CIU limit: 1"

2:54:31:758 - "Module Mode Control Disabled."

If for any reason something is not functioning properly, check the messages first as it is the Simulator's means of indicating that something went wrong. You may also wish to review the Microsoft application Event Log. Further details is provided in the Appendix.

Watch Tab



The Watch tab provides the means to monitor a number of Watch Block addresses. Simply type the Loop | PCU | Module | Block address into

the **Address** field, and watch the value change in the **Value** field. You can add a comment to the **Comment** field so that you can recognize the meaning of each monitored value.

Pause the watch function with the **Suspend Watch** checkbox.

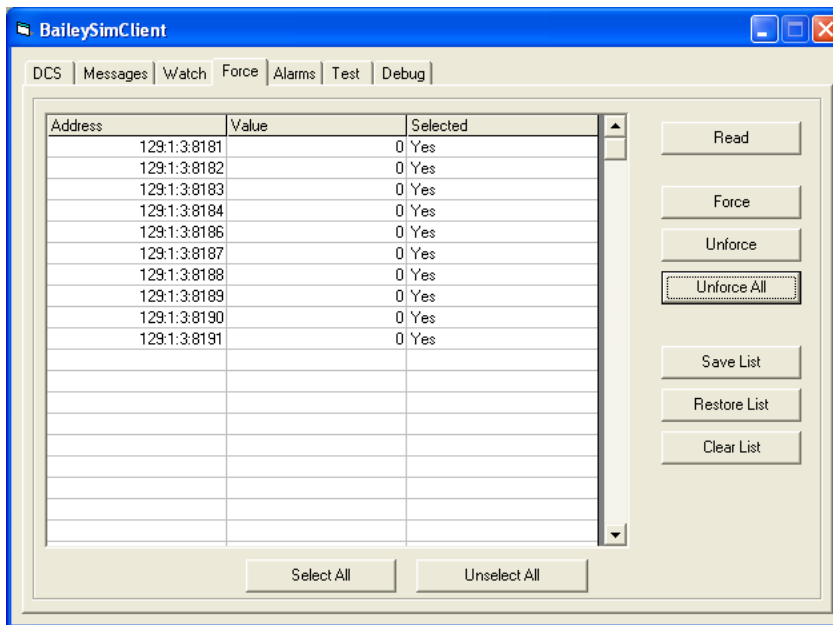
Save and restore the Watch Block list with the **Save List** and **Restore List** functions.

Watch Performance Statistics

The following address strings, when specified as an address in the Watch tab functions, support specific performance measurements

- **PerfCpu** – When this *address* is selected, the *value* shown is the average percentage of CPU usage by the simulator for 100 cycles of calculations, where ONE cycle is defined via OPTIONS line parameter E in the default.INI file.
- **PerfSlip** - When this *address* is selected, the *value* shown is the average slip in milliseconds per second calculated over 100 cycles of calculations, where ONE cycle is defined via OPTIONS line parameter E in the default.INI file.
- **<L>:<P>:<M>:16** – When this valid address is selected, the value provided is the CPU utilization time for the selected module. This is similar to output N+2 (Elapsed time of current cycle in units set by S1) of FC82 - Segment control block.

Force Tab



The Force Tab supports forcing an array of block addresses to the specific values entered at this table. This means, for example, that if a particular Boolean block output is set to a Force Value of 1, then this value will

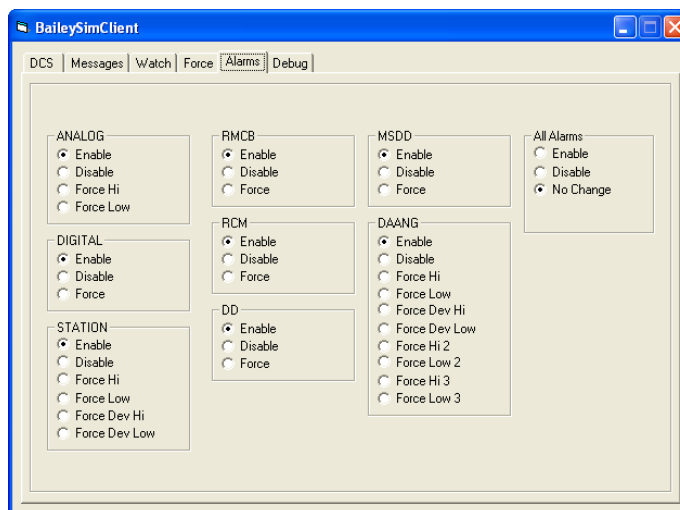
remain set to 1 regardless of the upstream logic that would otherwise affect that block address. The controls available are:

- **Read** – Read the current actual value at the selected addresses.
- **Force** – Force the selected addresses to the value provided.
- **Unforce** – Release the selected addresses to assume the value determined within the control logic.
- **Unforce All** – Remove all force block settings, including those not in the current display list. You will be prompted to confirm you wish to continue with the dialogue: *“This action will unforce all blocks in all modules. Proceed?”*.
- **Save List** – Save the current list to file, to restore later on.
- **Restore List** – Restore a selected list of force values.
- **Clear List** – Clear the current display list
- **Select all** – Select all table entries
- **Unselect all** – Unselect all table entries
- **Table entry** – Enter the address that you wish to force, the value to force it to, and click the Force button to apply.

NOTE: Forced values are saved to the DCS state files if in effect at time of DCS file save. These forced values are then restored at DCS file restore. If you wish to remove these forced values from a DCS state file you must (a) restore the DCS state file, (b) clear the forced values and (c) save the DCS state file again.

NOTE: The BaileySimClient will only show force values that have been created within the current session. There may be forced values present from previous sessions that do not show up in the list.

Alarms Tab



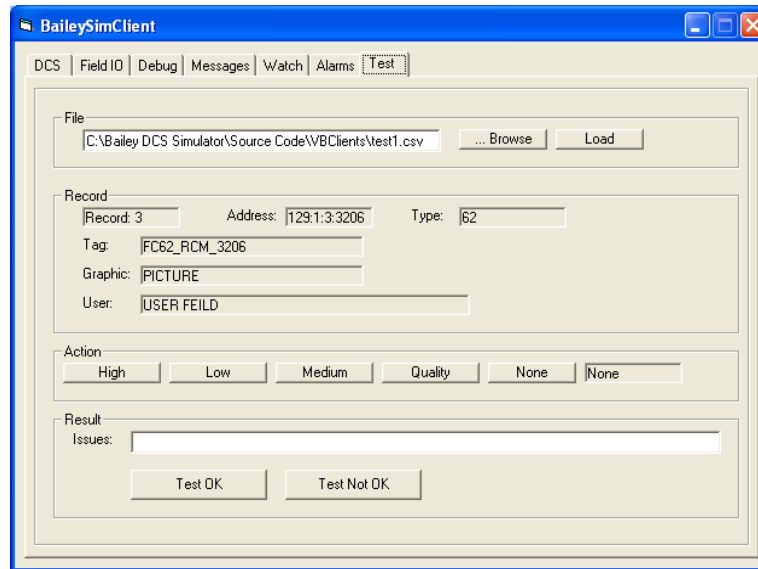
Using the Alarms tab the user can:

- DISABLE all alarms for all tag types
- ENABLE all the alarms for all tag types.
- Selectively ENABLE/DISABLE alarms by tag type
- FORCE alarms to any particular alarm state for all alarms of a given tag type

This functionality is provided to assist in testing the alarm system for any HMI connected to the Bailey DCS Simulator.

NOTE: Alarms work slightly differently for some blocks compared to others. For example, after you've FORCED the alarm state for RMCB blocks, you must first DISABLE and then ENABLE alarms to return to normal block operation. This is caused by the fact that some blocks retain Last Alarm State as an internal variable.

Test Tab



The Test tab is specifically to support testing and commissioning of an HMI connected to the Bailey DCS Simulator. The Test tab supports selection of a sequence of specific exception generating blocks and injection of a test signal at that block so that it may be verified that the correct screen element is affected by the injected signal.

NOTE: This tab is only available when the Bailey DCS Simulator is started in Loopback mode.

File

The file section supports selection of a CSV file of Test Signals. This file **MUST** have a specific CSV file format of records and fields. **Browse** to find the file you want and then **Load** it.

Each record corresponds to one **test case**. Each record must be formatted to contain the following fields:

- **Loop** – The loop number where the signal will be injected.
- **PCU** – The PCU number where the signal will be injected.
- **Module** – The module number where the signal will be injected.
- **Block** – The block number where the signal will be injected.
- **Tag** – the tag name corresponding to this address.
- **Graphic** – The graphic name upon which this address is referenced
- **User** – Free format user field

The user would normally automatically generate the test file from the base DCS EWS system files. A sample CSV file to illustrate the correct file

format is included within the Sample Test File directory within the CD-ROM. The Test File may be located at any directory.

Record

As the test progresses, the user will sequence through the test file from the first test case record, to the second test case record and so on through to the last test case record in the Test File.

At each record, this section of the screen displays the CSV file information for that record as follows:

- **Record** – Displays the record number in the CSV Test File.
- **Address** – Displays the L:P:M:B address from the CSV Test File
- **Type** – Displays the block type at the above address. It is expected that the user will verify that this type is correct.
- **Tag** – Displays the Tag Name. The tool doesn't act on this data, it only displays this for user reference.
- **Graphic** – Displays the Graphic name upon which this tag should appear. The tool doesn't act on this data, it only displays this for user reference.
- **User** - Displays the free form user field. The tool doesn't act on this data, it only displays this for user reference.

Action

To test a given test case it is assumed that the user will bring up a graphic display on an operators console (not part of the Bailey DCS Simulator). Then, while watching the HMI display, the user will use the buttons in this screen region to inject a signal into the corresponding L:P:M:B address. The user should see consequential change at the appropriate screen element, connected via the correct tag name, on the correct graphic.

The signals available for connection are:

- **High** – Inject some “High” signal if possible.
- **Low** – Inject some “Low” signal if possible
- **Medium** – Inject some “middle of range” signal if possible
- **Quality** – Inject Bad Quality signal.
- **None** – No signal is injected. Any previous injected signals are removed. The target block returns to its normal behavior.

Test signals are currently supported for each of the following function code types: FC30, FC45, FC62, FC68, FC80, FC123, FC129, FC136, FC151, FC177. If you require support to test any other exception generating function code type please contact Previs.

Multiple signals are provided simply to give the user the ability to find one signal that generates a clearly visible reaction on the HMI screen

Result

Once the test signal is injected, the user must then decide whether of not this specific test PASSED or FAILED.

If the Test PASSED then the user should click the TEST OK button. This will cause a TEST OK record to be added to the Test Result File. A click on this button also causes the record counter within the Test File records to increment so that the Test tab now points to the next test record.

If the Test FAILED then the user should click the TEST NOT OK button. This will cause a TEST NOT OK record to be added to the Test Result File. A click on this button also causes the record counter within the Test File records to increment so that the Test tab now points to the next test record.

If the Test FAILED then the user may also enter a free form text string at the Issues field. This data will be written to the Test Result file for this test record.

Sample Test Sequence

The typical test sequence is:

1. Make the CSV Test File.
2. Sort the records to assist in test sequencing as well as possible.
3. In the Test tab, **browse** to select the CSV Test File you want and **load** it.
4. Review the first test record.
5. Navigate to the correct HMI graphic and locate the screen object that “should” be connected to this test record.
6. Inject a signal at the Test Record block address.
7. Note whether the right screen object changed or not.
8. Record test pass or failure
9. Move to next Test Record
10. Repeat until complete.

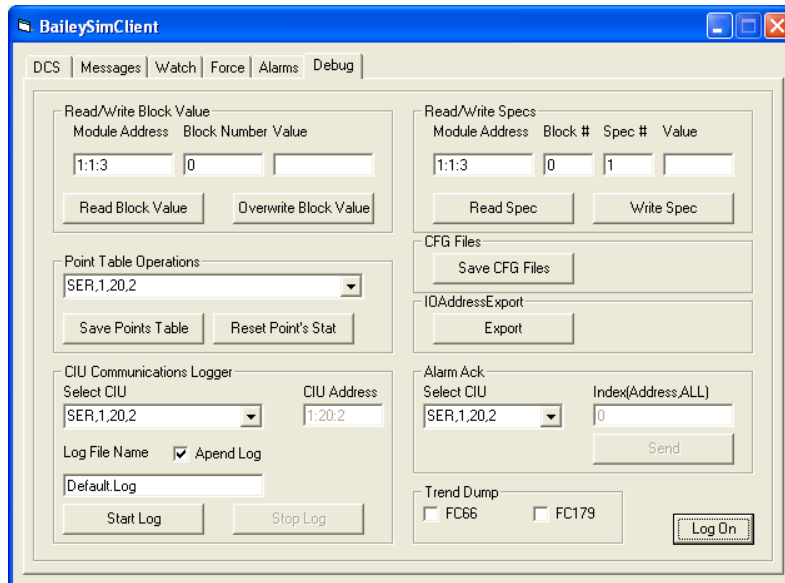
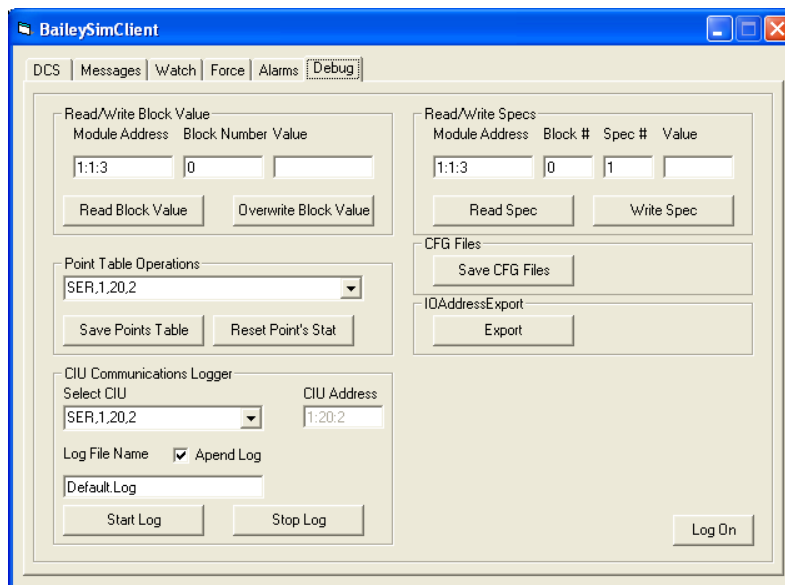
Test Result File

The Test Result file is created at the same directory path at which the Test File is located and contains all field from the Test File plus two additional fields:

- **Test Result** – TEST OK or TEST NOT OK as selected at the Result section.

- **Test Result Note** – The free form text entered at Issues.

Debug Tab



Read/Write Block Value

Read and write block values.

Read/Write Specs

Read and write block specifications.

Point Table Operations.

Save Points Table will save, to a CSV file, the console points list for one selected CIU. This is useful as a means to detect address mismatches between the console tag database and the CFG files loaded into the simulator.

Reset Points Statistics will reset the exception counters for the exception statistics section of the Points Table Report.

Points Table CSV File Format

The format of the Point Table file is provided in the Appendix.

CIU Communications Logger

This function is protected and is for Previsi use only and is not documented here.

Save CFG File

Causes all CFG file to be saved so that any tuning or other changes are not lost at startup.

IO Address Export

This function is intended to produce, to the best quality available from within the CFG files, a list of all IO that would need to be included in the API read/write arrays. Contact Previsi if you feel you need this function.

Log On

Permits password protected access to restricted debug features for test of certain Alarm ACK and trend functions. After log on, the Debug Tab presents hidden features Alarm ACK and Trend Dump.

Alarm ACK

(Restricted access) Acknowledge one tag by *index* or by *L:P:M:B address*, or acknowledge all tags typing ALL in the edit box. Alarm ACK directly ONLY to console connected to selected CIU.

Trend Dump

(Restricted access) Allows dump all communication to trend blocks into the log file in an easy to understand form.

Appendices

Appendix A – Function Code Support

The following table lists the supported function codes. Contact Previsé if you may require additional FC support.

Supported Function Codes					
FC	Function Code Description	Supported	Notes		
			3	8	Other
1	Function Generator	Yes			
2	Manual Set Constant (Signal Generator)	Yes			
3	Lead/Lag	Yes			
4	Pulse Positioner	Yes			
5	Pulse Rate	Yes			
6	High/Low Limiter	Yes			
7	Square Root	Yes			
8	Rate Limiter	Yes			
9	Analog Transfer	Yes			
10	High Select	Yes			
11	Low Select	Yes			
12	High/Low Compare	Yes			
13	Integer Transfer	Yes			
14	Summer (4-Input)	Yes			
15	Summer (2-Input)	Yes			
16	Multiply	Yes			
17	Divide	Yes			
18	PID Error Input	Yes			
19	PID (PV and SP)	Yes			
20	Indicator Station	Yes	X		
21	M/A Station (Basic)	Yes			
22	M/A Station (Cascade)	Yes			
23	M/A Station (Ratio)	Yes			
24	Adapt	Yes			
25	Analog Input (Same PCU Node)	Yes			
26	Analog Input/Loop	Yes			
27	Analog Input	Yes			
28	Analog Output (Same PCU Node)	Yes			
29	Analog Output	Yes			
30	Analog Exception Report	Yes			
31	Test Quality	Yes			
32	Trip	Partial			10
33	Not	Yes			
34	Memory	Yes			
35	Timer	Yes			
36	Qualified OR (8-Input)	Yes			
37	AND (2-Input)	Yes			
38	AND (4-Input)	Yes			
39	OR (2-Input)	Yes			
40	OR (4-Input)	Yes			
41	Digital Input/Controlway/Module Bus	Yes			
42	Digital Input/Loop	Yes			
43	TCS Digital Input	Yes			
44	TCS Digital Output	Yes			
45	Digital Exception Report	Yes			

Supported Function Codes

FC	Function Code Description	Supported	Notes		
			3	8	Other
46	Digital Input List	Pending			
47	Analog Input Exception Report (NAMM01)	No			2
48	Reserved	Not applicable			
49	Digital Output Buffer	Partial			10
50	Manual Set Switch	Yes			
51	Manual Set Constant	Yes			
52	Manual Set Integer	Yes			
53	Executive Block (COM)	Yes			
54	Executive Block (NLMM01)	No			2
55	Hydraulic Servo	Yes		X	
56	Executive Block (NAMM01)	No			2
57	Node Statistics Block (HAC)	Partial			10
58	Time Delay (Analog)	Yes			
59	Digital Transfer	Yes			
60	Group I/O Definition	Pending			
61	Blink	Yes			
62	Remote Control Memory	Yes			
63	Analog Input List (Same PCU)	Yes			
64	Digital Input List (Same PCU)	Yes			
65	Digital Sum With Gain	Yes			
66	Analog Trend	Yes			5
67	Reserved	Not applicable			
68	Remote Manual Set Constant	Yes			
69	Test Alarm	Yes			
70	Analog Point Definition	Yes			
71	Executive Block NAMM02/IMAMM03	Yes	X		
72	Analog Slave Definition	Yes			
73	Calibration	Yes			
74	Calibration Command	Pending			
75	Analog Calibration Status	Pending			
76	Thermocouple Temperature	Pending			
77	Analog Point Service Status	Yes			
78	Trend Definition	Yes			
79	Control Interface Slave	Yes			
80	Control Station	Yes			
81	Executive	Yes	X		1,5
82	Segment Control	Yes	X		1,5
83	Digital Output Group	Yes			
84	Digital Input Group	Yes			
85	Up/Down Counter	Yes			
86	Elapsed Timer	Yes			
87	DLS Interface	Yes			
88	Digital Logic Station	Yes			
89	Last Block	Yes			
90	Extended Executive	Yes	X		1,5
91	BASIC Configuration (MFC/MFP)	Yes			7
92	Invoke BASIC	Yes			7
93	BASIC Real Output	Yes			7
94	BASIC Boolean Output	Yes			7
95	Module Status Monitor	Yes	X		1,5
96	Redundant Analog Input	Yes			
97	Redundant Digital Input	Yes			
98	Slave Select	Partial			10
99	Sequence of Events Log	Yes	X		
100	Digital Output Read Back Check	Partial			10
101	Exclusive OR	Yes			
102	Pulse Input/Period	Partial		X	10
103	Pulse Input/Frequency	Yes		X	
104	Pulse Input/Totalization	Yes		X	11

Supported Function Codes					
FC	Function Code Description	Supported	Notes		
			3	8	Other
105	Executive Block (IMLMM02)	Yes			
106	Segment Control Block	Yes			
107	Group I/O Definition (IMLMM02)	Yes		X	
108	Extended Executive (IMLMM02)	Yes			
109	Pulse Input/Duration	Yes		X	
110	Rung (5-Input)	Yes			
111	Rung (10-Input)	Yes			
112	Rung (20-Input)	Yes			
113	ASCII String Descriptor	Pending			
114	BCD Input	Partial			10
115	BCD Output	Partial			10
116	Jump/Master Control Relay	Partial			10
117	Boolean Recipe Table	Yes			
118	Real Recipe Table	Yes			
119	Boolean Signal Multiplexer	Yes			
120	Real Signal Multiplexer	Yes			
121	Analog Input/INFI-NET	Yes			
122	Digital Input/INFI-NET	Yes			
123	Device Driver	Yes			
124	Sequence Monitor	Yes			
125	Device Monitor	Yes			
126	Real Signal Demultiplexer	Yes			
127	Plant Loop Gateway Node Map	Pending			
128	Slave Default Definition	Yes		X	
129	Multi-State Device Driver	Yes			
130	Plant Loop Gateway Executive	No			2
131	Plant Loop Gateway Point Definition	No			2
132	Analog Input/Slave	Yes			
133	Smart Field Device Definition	Yes	X		
134	Multi-Sequence Monitor	Partial			10
135	Sequence Manager	Yes			
136	Remote Motor Control	Yes			
137	C and BASIC Program Real Output With Quality	Yes			7
138	C or BASIC Program Boolean Output With Quality	Yes			7
139	Passive Station Interface	Yes		X	
140	Restore	Yes			5
141	Sequence Master	Partial			10
142	Sequence Slave	Partial			10
143	Invoke C	Yes			7
144	C Allocation	Partial			10
145	Frequency Counter/Slave	Yes		X	
146	Remote I/O Interface	Yes		X	
147	Remote I/O Definition	Yes		X	
148	Batch Sequence	Yes			1, 5, 9
149	Analog Output/Slave	Yes		X	
150	Hydraulic Servo Slave	Partial			10
151	Text Selector	Yes			
152	Model Parameter Estimator	Partial			10
153	Parameter Converter	Partial			10
154	Adaptive Parameter Scheduler	Partial			10
155	Regression	Partial			10
156	Advanced PID Controller	Yes			
157	General Digital Controller	Yes			
158	Enhanced Analog Point Definition	Yes			
159	Polynomial Adjustment	Yes			
160	Inferential Smith Controller	Yes			
161	Sequence Generator	Yes			
162	Digital Segment Buffer	Yes			
163	Analog Segment Buffer	Yes			

Supported Function Codes					
FC	Function Code Description	Supported	Notes		
			3	8	Other
164	Segment Control (IMCOM04)	No			2
165	Moving Average	Yes			
166	Integrator	Yes			
167	Polynomial	Yes			
168	Interpolator	Yes			
169	Matrix Addition	Partial			10
170	Matrix Multiplication	Partial			10
171	Trigonometric	Yes			
172	Exponential	Yes			
173	Power	Yes			
174	Logarithm	Yes			
175	Sequence Executive	Pending			
176	Sequence Station (CSC)	Pending			
177	Data Acquisition Analog	Yes			
178	Data Acquisition Analog Input/Loop	Yes			
179	Enhanced Trend	Yes			1,5
180	Batch Input/Output	Pending			
181	Batch Station (CBC)	Pending			
182	Analog Input Definition	Pending			
183	Batch Executive (CBC)	Pending			
184	Factory Instrumentation Protocol Handler	Partial			10
185	Digital Input Subscriber	Partial			10
186	Analog Input Subscriber	Partial			10
187	Analog Output Subscriber	Partial			10
188	Digital Output Subscriber	Partial			10
189	Reserved	Not applicable			
190	User Defined Function Declaration	Partial			4, 10
191	User Defined Function One	Partial			4, 10
192	User Defined Function Two	Partial			4, 10
193	User Defined Data Import	Partial			4, 10
194	User Defined Data Export	Yes			1, 5, 9
195	Reserved	Not applicable			
196	Reserved	Not applicable			
197	Reserved	Not applicable			
198	Auxiliary Real User Defined Function	Partial			4, 10
199	Auxiliary Digital User Defined Function	Partial			4, 10
200	INFI-NET to Plant Loop Local Transfer Module Exec Blk (INIPT01)	Pending			
201	Data Point Definition	Pending			
202	INFI-NET to INFI-NET Remote Transfer Module Exec Blk (INIIT02)	Pending			
203	INFI-NET to Plant Loop Remote Transfer Module Executive	Pending			
204	Reserved	Not applicable			
205	Reserved	Not applicable			
206	Reserved	Not applicable			
207	Reserved	Not applicable			
208	Reserved	Not applicable			
209	Reserved	Not applicable			
210	Sequence of Events Slave	Partial			6, 10
211	Data Acquisition Digital	Partial			10
212	Data Acquisition Digital Input/Loop	Partial			10
213	Reserved	Not applicable			
214	Reserved	Not applicable			
215	Enhanced Analog Slave Definition	Yes		X	
216	Enhanced Analog Input Definition	Yes		X	
217	Enhanced Calibration Command	Yes		X	
218	Phase Execution	Pending			
219	Common Sequence	Yes			1, 5, 9
220	Batch Historian	Yes			10
221	I/O Device Definition Input Specifications	Yes	X		
222	Analog In/Channel	Yes	X	X	

Supported Function Codes					
FC	Function Code Description	Supported	Notes		
			3	8	Other
223	Analog Out/Channel	Yes	X	X	
224	Digital In/Channel	Yes	X	X	
225	Digital Out/Channel	Yes	X	X	
226	Test Status Block Output	Yes	X		
227	Gateway (Harmony)	Yes	X		
228	Foreign Device Definition	Yes	X		
229	Pulse In/Channel	Yes	X	X	9
230	Strategic Loop Controller I/O	Pending			
231	Strategic Loop Controller Station	Pending			
232	Reserved	Not applicable			
233	Reserved	Not applicable			
234	Reserved	Not applicable			
235	Reserved	Not applicable			
236	Reserved	Not applicable			
237	Reserved	Not applicable			
238	Reserved	Not applicable			
239	Reserved	Not applicable			
240	Reserved	Not applicable			
241	DSOE Data Interface (Harmony)	Yes	X		6
242	DSOE Digital Event Interface	Yes	X		6
243	Executive Block (INSEM01)	Yes	X		6
244	Addressing Interface Definition	Yes	X		6
245	Input Channel Interface	Yes	X		6
246	Trigger Definition	No			
247	Condition Monitoring	Yes	X		

Notes

1. Some intentional differences from actual Bailey Function Code behavior to support simulator behavior.
2. This is an obsolete Function Code or this function code is not likely to be supported for this, or some other reason.
3. Supported only to the extent to allow the simulator to run. No internal function is implemented, nor is any needed for normal simulator operations.
4. This is an application specific function code, and will be implemented on a case by case basis if needed.
5. There may be some variance from native behavior.
6. Sequence of Events is not supported. However digital inputs for SOE system are supported, with digital values set via API.
7. These function codes associated with C and BASIC program segments have been implemented in such a manner that the C or BASIC function is executed EXTERNAL to the Bailey DCS Simulator, and IO with the simulator is handled via the API provided for that purpose.
8. These Input/Output blocks are implemented, but in such a manner that there is no field IO required. All actual IO to the control logic is via READ/WRITE to the Bailey DCS Simulator via the API provided for that purpose.
9. There are known issues with the implementation of this block. Contact Previsé if you require use of this.
10. An empty shell has been implemented for this block, and if encountered in your control logic this block will show as “under construction”. However, the rest of your CFG file will work, and you can write to the outputs of this block via the API or BaileySim client. We'll implement internal function of these blocks if and when needed only.

11. This block supports field IO input written via API in counts per second, so that other block functions are maintained. Refer to DCSIOManager description in appendix K.

If a Function Code is Not Supported

If a Function Code in a controller CFG file is not supported, the CFG file will still load and execute within the Simulator. A list of the unsupported Function Codes will be provided at the Microsoft Event Log and at the Debug tab of the BaileySimClient. Block outputs for the unsupported block will be 0 with good quality (Q).

Getting Previsе to Review your Requirements

If you are interested in the Bailey DCS Simulator, but you think you may require further Function Code support, contact Previsе and we can assist you in this review at no cost.

Appendix B – Configuring the *Default.INI* File

Default.INI Configuration File

You may use the *Default.INI* file to configure the DCS structure in the Bailey DCS Simulator. The *Default.INI* file is located at the *..Program Files\Bailey DCS Simulator\..* path.

Each line in the Default.INI file, except comment lines, is used to configure some aspect of the Bailey DCS to be simulated. The following line types are supported:

- * - Comment. This line is not executed
- OPTIONS – general simulator parameters
- BESC – Exception Reporting configuration
- LOOPBACK – Run in loop back mode, with parameters
- EVENTS – Set log verbosity
- IOMANAGER – Start DCSIOManager
- CONTROLLER - Set up controller module (replaces MOD)
- CIU – Set up CIU (Computer interface unit) module
- CIUCHANNEL Set up CIU (new – for Composer connection)
- TMST – Set up simulated time synchronization master
- ALARMS – Configure alarm acknowledgement

All line types are optional. No entries are mandatory.

*** – Comment**

You can add comments to your configurations or comment things out by adding an * at the beginning of the line. The whole line will be disregarded in the INI processing.

OPTIONS – General Options Switch

The OPTIONS line supports several options switches.

Format *OPTIONS,A,B,C,D,E,F*, where:

- A - CFG File Save Option (0 - NOT SAVED, 1 - SAVED)
- B - Execution startup mode (0 - EXECUTE, 1 - PAUSE)
- C – Path for DCS state files (should be on local computer)
- D – Minimum segment execution period in integer milliseconds
- E – Simulator Execution Period in integer milliseconds
- F – Batch 90 Emulation (0 - DISABLE, 1 – ENABLE)

CFG File Save Option

If set to NOT SAVED then the CFG files are not saved and any changes made to controller logic (i.e. tuning, CFG upload etc) will not be saved and will be lost at simulator shutdown.

If set to SAVED, then CFG files will be saved at the following events:

- Affected single CFG file will be saved at any tuning change via console or EWS
- Affected single CFG file will be saved at any CFG file upload from EWS to simulated DCS
- All CFG files will be saved at simulator shutdown.

This default.INI setting does not affect the SaveCFGFile function when issued to the simulator via the application programming interface. Regardless of this setting, this API command will cause CFG files to be saved.

CFG files will be saved at the path defined in the default.INI file. If you wish to add a new module you must first define it in the default.INI file and place the initial CFG file at the defined CFG directory path.

Execution Startup Mode

If set to EXECUTE then execution of all controller logic will commence automatically at startup.

If set to PAUSE then execution of all controller logic will pause immediately at startup, awaiting a command to resume execution via the application programming interface or via the BaileySim Client

Path for DCS State Files

Set the path for DCS state files save and restore. If no path is defined the Bailey DCS Simulator Program Files path is used.

NOTE: It is recommended that a path for DCS state files be defined so that these files collect by themselves in their own directory.

Minimum Segment Execution Period

Normally Function Code 82 (Segment Control) Specification S2 is used to specify how often the segment is to be executed. For example if S2 is set to 0.250 seconds, then this segment will be executed every 250 milliseconds at a 1 X real time throttle setting.

This causes a problem if S2 is set to very small numbers, because of system CPU loading and a Microsoft operating system limitation that processes can only be scheduled at integer multiples of 10 milliseconds.

If no option is specified for minimum segment execution time, then a hard coded minimum execution time of 250 milliseconds is used within the Bailey DCS Simulator, and no segment will execute more frequently than this, regardless of actual S2 settings.

If any specification S2 is set to a execution frequency that is NOT a multiple of 10 milliseconds, then the segment execution frequency is rounded up or down to the nearest multiple of 10 milliseconds.

If the option is specified for minimum execution frequency then the following rules apply:

- The default minimum segment execution frequency of 250 milliseconds is overridden by the OPTIONS setting
- Any option set for minimum execution frequency faster than 20 milliseconds will be set to 20 milliseconds. In other words the simulator will not support segment execution faster than once every 20 milliseconds.
- If the option for execution frequency is set to a number that is not a multiple of 10 milliseconds, then the above rounding rule applies.
- Low parameter values may have a big effect on CPU load. You may need to experiment with different values to achieve a tradeoff between faster operation and acceptable CPU load.

Simulator Execution Period

The Simulator Execution Period (jn milliseconds) defines the simulator “clock tick”, the interval at which module segments are actually executed.

The Bailey DCS Simulator has it's own module execution clock, execution counter(T_{ec}) in milliseconds. When the Bailey DCS Simulator starts up, the Execution Counter Timer is set to 0 ($T_{ec} = 0$).

Every time the Execution Counter Timer expires (reaches 0):

- Each segment of each controller module defined and loaded at startup is “executed” as described below.
- Reset the Execution Counter Timer: $T_{ec} = T_{ec} + T_{ep}$. The Execution Counter Timer is reset to the Execution Period(T_{ep})
- Then wait until the Execution Counter Timer expires again.

The default value for Simulator Execution Period is 250 milliseconds. The minimum acceptable value for Simulator Execution Period is 50 milliseconds. If any value < 50 milliseconds is entered for Simulator Execution Period then the default value of 250 milliseconds is used.

Generally there is no reason to use any value other than the 250 millisecond default value for this parameter.. Larger values may result in unexpected changes to logic behavior. Smaller values may not result in any improvement in behavior.

Execution of a Controller Module Segment

This describes how a module segment is executed:

- If controller module is not in EXECUTE mode, it is not executed. Each module segment has it's own time counter T_s , set to 0 when the module switches to EXECUTE mode

- When invoked for execution, the module receives the current Execution Counter Timer T_{ec} and then proceeds to executes all segments within the module in sequence.
- Each segment has an Segment Execution Counter Timer T_{iec} , which is set to T_{ec} , when module switches to EXECUTE mode
- Each time the segment is executed:
 - Segment time is increased by period defined in FC82 spec S2. (i.e. $T_s = T_s + FC82 - S2$)
 - If $T_s < T_{ec} - T_{iec}$, then
 - Segment is executed
 - $T_s = T_s + FC82 - S2$
 - Repeat
 - If $T_s > T_{ec} - T_{iec}$,
 - execution is finished

The end result is that:

- The Simulator Execution Period defines a simulator “clock tick”
- Module segments with segment execution period less then the Simulator Execution period will execute one or more times each clock tick
- Module segments with segment execution period greater than Simulator Execution will execute when the time comes up.

Batch 90 Emulation

The Batch 90 switch needs to be set if you wish to execute Batch 90 programs within the Bailey DCS Simulator. In addition, you must ensure that your license for the Bailey DCS Simulator supports Batch 90.

NOTE: *To access the Batch 90 functions within the Bailey DCS Simulator, your license must include Batch 90 Support.*

If you ENABLE Batch 90 emulation via the switch provided (0 = DISABLE, 1 = ENABLE), and if your license supports Batch 90, then the Bailey DCS Simulator will look for the file Batch90.INI in the **..ProgramFiles\Bailey DCS Simulator\..** directory, and will configure the Batch functions according to the instructions in this file.

Refer to the manual entitled **Users Manual, Bailey DCS Simulator, Batch Addendum** for further details about use and operation of the Bailey DCS Simulator Batch 90 functions.

BESC – Exception Reporting configuration

The BESC line is used to create a multiplicative scale factor for Specification S8 for all Segment Control Blocks FC82. FC82 Spec S8 is the Maximum Exception Report Period, and defines what is normally referred to as the “background” exception reporting interval.

A *Background Exception Screening Mode* is also available

In large simulator configurations, with many tags, console communications can be improved by suppressing or screening the background exceptions using the BESC configuration options.

NOTE: BESC line MUST follow immediately after the OPTIONS line.

Format ***BESC,X,A,B,C,D***, where:

- X - leave this parameter alone
- A - Scaling factor ≥ 1
- B - Maximum resulting value for FC82 Spec S8
- C - Minimum resulting value of FC82 Spec S8
- D – Background exception screening mode

Example:

BESC,X,2,1000,60,2

This will scale all S8 by factor of 2 to maximum of 1000 and minimum of 60 (seconds), and will select background exception screening mode 2.

Background Exception Screening Mode

Background Exception Screening Mode is defined as follows:

0. All background exceptions are generated and sent to console.
1. Always send background exceptions to the console **ONLY** when they report data change (i.e. difference from last exception sent for this address).
2. IF console requests background exception screening (via bit in RESTART command) THEN send background exceptions to the console **ONLY** when they report data change (as in mode 1) ELSE send all background exceptions to the console
3. NEVER send background exceptions to console
4. IF console requests background exception screening THEN send **NO** background exceptions to the console ELSE send **ALL** background exceptions to the console

For all background exception screening modes, the timing of background exceptions, when they are sent, remains at interval defined by FC82-S08 and BESC Parameters A, B and C.

Using a combination of parameters the BESC line can be used to reduce the frequency of background exceptions and screen those that need not be sent, thus reducing console communications traffic. And all of this is accomplished without changing the controller CFG files.

Exception Statistics

Refer to the Points Table report defined elsewhere in this manual to obtain exception communication statistics. This may be useful to understand how to resolve communications issues.

LOOPBACK – Enable/Configure Loop Back Mode

To run the Bailey DCS Simulator in Loop Back Mode, insert the LOOPBACK line into the Default.INI file before definition of the modules.

Format **LOOPBACK,TA,TD,ES,1,1** where:

- **TA** – Period in seconds for simulating analog signals
- **TD** – Period in seconds for simulating digital signals
- **ES** – Switch to start generation of analog and digital signals immediately after Bailey DCS Simulator startup where 1 = start (Recommended for most installations) and 0 = suspend. If generation of signals is suspended, thus reducing the number of exceptions transmitted during console startup, it can be resumed manually at the *BaileySim* client. Recommend

Example of default.ini file

```
BESC,X,8,7200,300,0
LOOPBACK,60,100,0,1,1
MOD,MFC04,129,1,3,129_1_3.cfg
MOD,MFC04,1,5,6,1_5_6.cfg
MOD,MFC04,1,1,5,1_1_5.cfg
*MOD,MFC04,129,6,5,C:\validation_cfg_for_simulator\validate_tx1.cfg
* module at address 129:1:3 with configuration AIIcFs.CFG
CIU,TCP,20,0,0,xxxx
* Telnet CIU's will use loop 20
* CIU,ATL,1,3,2,xxxx
* ATL CIU at address 1:3:2
```

In this example TA = 60 S, TD = 100 S, initial generation of the signals is suspended.

EVENT – Set level of Event Log Verbosity

The Bailey DCS Simulator logs all significant events to the Microsoft Event Log. The BaileySim Client gets these events messages for the message tab.

The EVENT line in the default.INI file allows the user to define log verbosity level, to reduce the quantity of messages to the event log if certain messages happen frequently and are not required for troubleshooting. This increases the time period for which messages can be stored in the Microsoft Event Log file.

Format ***EVENTS,A,B,C,D,E*** where

- A = General verbosity (1=BRIEF, 2=VERBOSE=ALL)
- B,C,D,E = spare

VERBOSE (=ALL) means all events are logged. BRIEF means that the following specific events are not logged:

- CIU Messages suppressed at Verbosity 1 (Brief) are:
 - CIU (x.x.x) Command Environment (x.x)
 - CIU (x.x.x) Unsupported command xxxxxxxxxxxx
- General (non-CIU) events suppressed at Verbosity 1 (brief) are:
 - Execution paused
 - Execution Resumed
 - File <path>.cfg loaded into module x.x.x

In the following example BRIEF verbosity is selected:

EVENTS,1,B,C,D,E

IOMANAGER – Start DCSIOManager

This line starts DCSIOManager (see Appendix K) if required.

Format:

IOMANAGER,1,0,0,0,0

There are no options on this line. If the line is present, DCSIOManager is started. If the line is not present, DCSIOManager is not started

MOD - Set up Controller Module (OBSOLETE)

NOTE: The MOD line is obsolete, and has been replaced with the CONTROLLER line. If you currently use the MOD line, change your default.INI file to use the CONTROLLER line. Do not use the MOD line for new installations.

A MOD Line adds a controller module to the DCS configuration.

The format is:

MOD,<Type>,<L>,<P>,<M>,<filename.cfg>

Where:

- Type = Enter MFC04 for this field. NOTE: All module types are emulated via a common emulation. Your controllers are supported regardless of whether MFCxx, MFPxx or others.
- L - Loop Address
- P – PCU or node address
- M – Module address
- Filename.cfg – Path and filename to CFG file for this module.

CONTROLLER – Set up Controller Module

A CONTROLLER line adds a controller module to the DCS configuration.

The format is:

```
CONTROLLER,< Type>[:<Version>],<L>,<P>,<M>,[<path>]
```

Where:

- **Type** = The controller type actually used in your real DCS installation., selected from the list at Table B-1.
- **Version** – This is the firmware version of this controller. This is not used within the simulator logic, except within the module status returns provided to consoles and engineering tools.
- **L** - Loop Address
- **P** – PCU or node address
- **M** – Module address
- **Path** – Path, including full filename to CFG file for this controller. This parameter is only accepted for module types which require a CFG file load at startup (e.g. *10205.cfg*).

Example:

```
CONTROLLER,BRC100:A1,1,64,2,C:\project\L6402.cfg
```

```
CONTROLLER,IMMFC04:A2,9,3,6,c:\cfg_directory\90306.cfg
```

The second line would configure an MFC04 at address 9:3:6 and would cause the file 90306.cfg to be loaded at Simulator startup.

NOTE: All module types are emulated via a common emulation. This means that function codes have the same function regardless of what controller type you select. The controller type must nonetheless be correct, so that simulator status returns to engineering tools are correct.

Table B-1 – Module Type Table for CONTROLLER Line

Controllers			CIU modules	PCU modules
IMMPC01	IMMFP01	NCOM02	INICT01	INNPM01
	MFP01	IMCOM03	INICT03	INNIS01
SLC01	IMMFP02	COM03	IMCPM01	INBIM01
SLC21	MFP02	COM03A	IMCPM02	INBIM02
	IMMFP03	IMCOM04	IMCPM03	INLIM03
NMFC01	MFP03	COM04	IMSPM01	
NMFC02	IMMFP04	COM04A	IIMCP01/IIPST01	
MFC02	MFP04	IMLMM02	For CIU modules, the module address must be 2	For PCU modules, the module address must be 0
IMMFC03	IMMFP12	LMM02		
MFC03		NAMM02	Use exact syntax as noted in this table	
IMMFC04	BRC100	AMM02		
MFC04	BRC200	IMAMM03		
IMMFC05	BRC300	NAMM03		
MFC05	BRC400	AMM03		
		IMQRC01		

CIU – Set up CIU module

SCSI Device ID

Each SCSI channel consists of a SCSI Host, sometimes called SCSI Initiator, and 1 to 8 SCSI Targets. The Host (at the operator console end) is the communications master. The Target (the emulated INICT03 and any other devices in the SCSI channel) is a communications slave.

For example a SCSI Host may have several SCSI Devices attached in a SCSI daisy chain:

e.g.

- Device 0 – Hard disk
- Device 1 – hard disk
- Device 3 – MCP02
- Device 6 – Floppy
- Device 7 - Peripheral

Because each SCSI channel may contain several SCSI devices, and because the Device ID for each device MUST be unique, the SCSI Device ID selected for the CIU must be selected carefully. This address MUST not conflict with any other SCSI device ID address within the channel.

Guidance to Select Device ID

If the emulated ICT03 is to replace a real MCP02 or ICT03 then select the Device ID from the actual MCP02 or ICT03 being replaced. Refer to Appendix for instructions to locate Device ID in MCP02 or ICT03.

If the emulated ICT03 is not replacing a real ICT03 or MCP02 then you'll have to select the Device ID yourself.

How will you determine what unique Device ID addresses are available?

In Bailey systems the MCP02 or ICT03 will “most often but not always” be at Device ID 3 or 4. So you may wish to try Device ID 4 and then Device ID 3 to see if there are signs of address conflict.

In Microsoft systems with SCSI channels present, you should be able to switch into the SCSI BIOS at system boot time (*on the HOST not the TARGET*) and see what SCSI devices have automatically been enumerated by the SCSI BIOS. If you can see this list, then you can simply select an unused address for the SCSI Device at the Target (with preference for Device ID 3 or 4).

Observe carefully at system boot to see if you observe any signs at all of Device address conflict.

NOTE: Use the CIUCHANNEL line to set up a CIU for Composer Connection. The CIU and CIUCHANNEL line will be merged at a later release.

The CIU line configures an emulated CIU (Computer Interface Unit) module. The format is:

CIU,<Type>,<L>,<P>,<M>,<parameter>

Where:

- Type = CIU Module type, selected from:
 - SER – selects CIU04 via serial RS232
 - ICT03 – selects ICT03 via serial RS232
 - ICT03:<FW> - selects ICT03 via serial RS232
 - TCP – Selects CIU04 via TCP/IP for connection by OPsCon and variants thereof.
 - SCSI:MCP02:<FW> – Selects first SCSI connection. Special SCSI hardware must be acquired from Previsé.
 - SCS2:MCP02:<FW> - Selects second SCSI connection. Special SCSI hardware must be acquired from Previsé. SCS2 is a PREFERRED alternative to RTST (below). Do NOT combine BOTH SCS2 and RTST.
 - RTST:MCP02:<FW> – Selects second SCSI connection. Special SCSI hardware must be acquired from Previsé. See additional information for setup of RTTgSCSI.ini file for this channel within these Appendices. **NOTE: This line will be made obsolete in next version. Do not use.**
 - FW = Firmware version that will be reported to console for this CIU (e.g. E0, F1, etc).
- L - Loop Address
- P – PCU or node address
- M – Module address (**Must be set to 2 for SCSI channels**)

NOTE: L:P:M address must be unique, and must NOT conflict with any other L:P:M address.

- Parameter =
 - COM<Port>:<baudrate>:<parity> for serial CIU
 - Port = COM port address
 - Baudrate = usually 19,200 (baud)
 - Parity = None (blank or N), Odd (O), Even (E)
 - SCSI<C>:<ID>:<LUN> for SCSI CIU
 - C = SCSI Channel 0 or 1. Set Channel Number = 0 for first SCSI channel & Channel Number = 1 for second SCSI channel.
 - ID = SCSI Device ID (0,1,2,3,4,5,6,7). Refer to sidebar instructions for guidance.
 - LUN = Logical Unit (**MUST be 0**)
 - “xxx” otherwise.

Examples:

```
CIU,SER,1,2,3,COM2:19200
```

This configures a CIU04 at DCS address 1:2:3 and routes this CIU through COM2: at 19200 baud.

```
CIU,ICT03,1,2,3,COM5:9600
```

This configures an INICT03 at DCS address 1:2:3 and routes this CIU through COM5: at 9600 baud.

```
CIU,TCP,2,0,0,xxxx
```

This will establish all OPsCon TCP/IP to be on Loop 2. The first TCP/IP connection will then define a CIU at address 2:1:0, the next at 2:2:0 etc. If this line is NOT included, then all TCP/IP connections from OPsCon will be established by default at Loop 13 instead.

```
CIU,SCSI:MCP02:F1,5,2,2,SCSI0:4:0
```

If the correct SCSI adapter from Previs is installed, this will establish an emulated MCP02 with firmware version F1 (same as ICT03-SCSI with F1 firmware) device within the channel. The CIU L:P:M address for this device will be 5:2:2. The SCSI Device ID for this device will be 4.

```
CIU,SCS2:MCP02:E0,5,1,2,SCSI1:3:0
```

If the correct SCSI adapter from Previs is installed, this will establish an emulated MCP02 with firmware version E0 (same as ICT03-SCSI with E0 firmware) device within the channel. The CIU L:P:M address for this device will be 5:1:2. The SCSI Device ID for this device will be 3.

CIUCHANNEL – Set Up CIU for Composer

NOTE: *Use the CIUCHANNEL line to set up a CIU for connection of ABB Composer to the Bailey DCS Simulator. The CIU and CIUCHANNEL line will be merged at a later release.*

The CIUCHANNEL line configures an emulated CIU (Computer Interface Unit) module. The format is:

```
CIUCHANNEL,<Type>[:FW],<L>,<P>,<Port>:<BaudRate>:<Parity>
```

Where:

- Type = CIU Module type, selected from:
 - ICT01 – selects ICT01 via serial RS232
 - ICT03 – selects ICT03 via serial RS232
- FW = Firmware version that will be reported for this CIU (e.g. E0, F1, etc)

NOTE: *For ICT03 set FW = E0*

- L - Loop Address
- P – PCU or node address

NOTE: Module address for CIU is set, by default, to 2. Address defined for CIU (L:P:2) MUST be unique and NOT conflict with any address.

- Port = COM port address (i.e. COM1)
- BaudRate = Selected baud rate for the <Port>
- Parity = None (blank or N), Odd (O), Even (E)⁵

Examples:

CIUCHANNEL,ICT03:E0,1,2,COM2:19200

This configures an ICT03, with firmware version E0, at DCS address 1:2:2 and routes this CIU through COM2: at 19200 baud and no parity.

. CIUCHANNEL,ICT03:E0,1,4,COM1:9600:E

This configures an ICT03, with firmware version E0, at DCS address 1:4:2 and routes this CIU through COM1: at 9600 baud with even parity.

TMST – Set Up Time Master

This line configures a simulated time master PCU within the DCS. This Time Master will not exist in your real DCS, but is added here to provide a simple means of synchronizing time between the Bailey DCS Simulator and all operator consoles.

The format is:

TMST,<ACC>,<L>,<P>,<WCO>,<TE>

- ACC= Clock accuracy. This must be set to be the highest priority within the system, so you may need to review Time Priority for each of the connected consoles. In most cases a value of 12, equivalent to a satellite clock, will be high enough.
- L = Loop Address for (simulated) Time Sync Master. Previsé suggests you select this to be the same Loop address as most controller modules (i.e. MOD line)
- P = PCU Address for (simulated) Time Sync Master. This must be a unique address, not used for any other CIU or MOD records elsewhere in the system.
- WCO = Wall clock offset time. This must be exactly 12 HEX characters and will need to be determined for your system. You'll need Previsé assistance to determine this.
- TE= Time Error in integer milliseconds (e.g. 21000, -14000). This will normally be set to 0 unless advised otherwise by Previsé.

Example

TMST,12,5,32,FFFECCBA3608,21000

This line creates a simulated time master of time priority 12 at loop 5, PCU 32. The time error of 21000 milliseconds will be added to the FFFECCBA3608 wall clock time offset and will be used to synchronize

⁵ To date, almost all consoles that have been connected to the Bailey DCS Simulator use NO parity. The only exception has been an OIS45 which required EVEN parity.

time to the consoles. When time synchronization is active, you will see an event logged to the Microsoft Event Log (and BaileySim client Message tab) like:

Time master Acc:12 Loop:5 Pcu:247 Tso:FFFECCB9E400 has been created.

Once you see this record, time to that console should be synchronized to within a second if the TMST line is set up correctly. Contact Previs if there are issues with this.

ALARMS

This line configures alarm acknowledgement. This feature is intended to assist with managing the alarm list within the consoles and supports:

1. All alarms can be disabled at simulator startup to prevent initial flood of alarms.
2. Alarm ACK on one console is shared to other consoles if alarm ACK share is supported by the connected console (currently supported for SuperLoop only).
3. A global alarm ACK function (i.e. ACK all alarms) is supported.

The format for the ALARMS line is:

ALARMS,A,B,C,D,E

- A – DISABLE (0) or ENABLE (1) all Alarms at startup of Bailey DCS simulator. If DISABLE is selected, no alarms will be generated until alarms are ENABLED, via API function call. Do not DISABLE alarms here unless (a) you do not want alarms at all or (b) you intend to ENABLE alarms via the suitable API call (Refer to Technical Manual - Bailey DCS Simulator API).
- B – DISABLE (0) or ENABLE (1) the Alarm ACK function. With this function DISABLED, no alarm ACK commands received from any console will be processed at all. With this function ENABLED, then alarm ACK commands received from any console will be forwarded to all other consoles that are currently online (applicable to SuperLoop systems only at present time).
- C – DISABLE (0) or ENABLE (1) the global alarm ACK function. If ENABLED then a global alarm ACK function is available and may be invoked by either:
 - Manual command at BaileySim Client
 - API command at Bailey DCS Simulator API (Refer to Technical Manual - Bailey DCS Simulator API).
- D – spare parameter
- E – spare parameter

Putting it all together

You can combine these configuration features in any way that makes sense in the Bailey DCS context. For example consider the following INI file contents:

```
BESC,X,2,1000,60,0
EVENTS,1,B,C,D,E
IOMANAGER,1,0,0,0,0
CONTROLLER,BRC100:C0,1,5,8,module.cfg
CIU,SER,1,2,2,COM2:19200
CIU,TCP,20,0,0,xxxxx
CIU,SCSI:MCP02:F1,5,2,2,SCSI0:4:0
CIU,RTST:MCP02:E0,5,1,2,SCSI1:3:0
CIUCHANNEL,INICT03:E0,13,3,COM1:19200
TMST,12,5,32,FFFECCEBA3608,21000
```

This INI file will:

- Decrease background exception traffic by a factor of 2,
- Selects BRIEF verbosity and starts DCSIOManager
- Add a BRC100 version C0 with module.cfg as module 1:5:8.
- Emulate an RS-232 CIU04 at address 1:2:2 and COM2:.
- Support OPsCon connections at Loop 20
- Emulate an SCSI MCP02(ICT03) at address 5:2:2, ID 4
- Emulate an SCSI MCP02(ICT03) at address 5:1:2, ID 3
- Emulates an ICT03, with firmware version E0, at DCS address 13:3:2 and routes this CIU through COM1: at 19200 baud.
- Emulate a Time Master of accuracy 12 at address 5:32

Appendix C – Event Log and Message Window Messages

The following table lists messages that are sent to the BaileySimClient Message tab page and to the Microsoft Application Event Log. The same messages are sent to both locations.

The Message tab will only show the messages that are new since the last time you started the Simulator.

Use the Microsoft Event Log Viewer to access the Application Event Log.

Event Log and Debug Window Messages	
Message	Interpretation
File Related Issues	
File Not Found	INI, CSV, CFG or DCS file not found
Error Reading CSV File Wrong Format in CSV file\n Error Reading CSV File	Issue with OPsCon tag database CSV file and should occur only if INI file includes @tagdatabase.CSV record
INI File Not Found Error Reading INI File Defaults File Not Found\n Error Reading Defaults File Invalid format in defaults file Invalid format in defaults file or invalid function code Unable to read default configuration default.ini	Issue with <i>default.INI</i> file
CFG File Issues	
File XX loaded into module <Loop>:<PCU>:<Module>	CFG file successfully loaded
Error Reading CFG File CFG file parse error: XX FC XX, Address XX File XX - has invalid spec format	Problems reading CFG file
Configuration file <path:filename>.cfg has been saved.	CFG file successfully saved by simulator.
State File Issues	
State <filename>.DCS has been restored.	Restore state file was successful
Finish read the header of state file <filename>.DCS..	Information only
State of the module - x:x:x has been restored..	State of one module was successfully restored
DCS file contains a non-existing module - L:P:M:.. FC at address L:P:M:B is no longer in configuration.. FC type at address L:P:M:B not the same as in DCS file.. State of FC at address L:P:M:B not in DCS file.. Wrong block type is defined for address - L:P:M:B (deleted) Error restoring FC from DCS file. Address: XX.	Messages indicate nature pf problem in restoring state file. Some problems are recoverable and others may not be. Restoration of state file can not be guaranteed for some CFG file changes. Issues likely caused by changing DCS configuration at Default.INI file or changing CFG files since DCS files was saved.
Module xx can not be restored completely. Configuration changed.. Unrecoverable error during restore of the module xx.	State not restored for this module
Failed to create file. Failed to open file. Error Writing DCS File Error Reading DCS File Error writing module details to DCS File	Issues with Save or Restore DCS file
DCS Revision # incompatible with this version of Simulator	DCS file may be linked to the version of the Simulator at time of DCS file save.
Service Status	
Service start and stop messages.	Service started. Service stopped.

Event Log and Debug Window Messages	
Message	Interpretation
Restart Simulator.	Simulator was restarted without stopping service.
USB License Key Issues	
No license key found or invalid license. Service Terminating	USB License key is required
Valid license found. Bailey DCS Simulator. Built: <date>. Block limit: XX Module limit: XX CIU limit: XX Unlimited Expiration Time License.. SCSI Enabled.. API Enabled.. Module Mode Control Enabled. Module Mode Control Disabled.	USB license key information messages
Can't add block. Block limit is XX Can't add Module. Module limit is XX. Can't add CIU. CIU limit is XX	Attempt to exceed license limits
CIU Connection Issues	
Serial Port COM1 19200 Ready. Serial Port COM1 9600 Ready.	Serial port ready to use
CIU(xx): Command Environment CIU(xx): Command Restart CIU(xx): Command Online/Offline CIU(xx): Command Read System Time CIU(xx): Command Read System Date and Time CIU(xx): Command Set System Date and Time CIU(xx): Command Set Time	Normal CIU communication messages
Reading serial port failed Write to serial port failed Comm inMsgSize < 1 Error setting com port mask Error setting com port mask Error setting com port timeouts Error setting com port settings Error opening com port Error creating CanSend event Error creating sender thread Error creating receiver thread	Issues with setting up and using serial RS-232 port for one of the simulated CIU connections. Verify all port settings and verify that ports are working. If problem persists contact supplier.
CIU(xx): Comm format error. CIU(xx): Comm error CIU(xx): Comm checksum error	Issues with setting up and using serial RS-232 port for one of the simulated CIU connections. Verify all port settings and verify that ports are working. If problem persists contact supplier.
New Telnet CIU address XX:XX:N Connection Accepted	At startup of TCP/IP connection by OPsCon client (or variant thereof)
Connection died	At termination of TCP/IP connection by OPsCon client (or variant thereof)
Telnet accept failed Unable to open start socket for TCP/IP communications	Internal problem.. Contact supplier. Occurs only when attempt to make TCP/IP connection from OPsCon or variant thereof
CIU(xx): Unsupported command: XX	CIU communications issue – contact supplier
Error creating CIU. Invalid CIU type. CIU has no PCU	CIU definition issues. Likely error parsing CIU record in default.INI file. Check syntax.
SCSI CIU Issues (Emulated INICT03)	
Inquiry CDB Received.	This message is received at console boot (BIOS level) or driver initialization. Several are likely to be received.

Event Log and Debug Window Messages	
Message	Interpretation
ADLL: <VScsiV2.dll > version: <#.#.#.> detected! SCSI<channel>:<target id>:<lun> Emulation running SCSI0:4:0 Emulation running. #1 - Emulation now running!	These are the messages you should see if the SCSI CIU channel 0 starts normally.
RTg(SCSI1:4:0): ADLL: VScsiV2.dll version: xx RTg(SCSI1:4:0): #1 - Emulation now running! RTg(SCSI1:4:0): #22. RTg(SCSI1:4:0): Inquiry CDB Received. RTg(SCSI1:4:0): RTTg SCSI1:4:0 Emulation running.	These are the messages you should see if the SCSI CIU channel 1 starts normally.
Failed to load SCSI DLL ADLL: GetFileVersionInfo failure! ADLL: hVersionInfoBuf allocation failure! ADLL: Get DLL version failed! ADLL: ERROR: minimum DLL version required: <#.#.#.> ADLL: LoadLibrary(<*.*.dll>) Failed GetProcAddress(<function name>) failed! VScsiV2.dll fail to activate" Unable to start SCSI emulation SCSI<channel>:<target id>:<lun> Failed to start #15 - DLL failed to process input data & establish emulation! #2 - Emulation channel has been Reset!" RTg(SCSI1:4:0): #2 - Emulation channel has been Reset!	Verify that the Bailey DCS Simulator is correctly installed, including all SCSI components. Review installation instructions in Users Manual and ensure all instructions followed. Verify that you can see the SCSI device within the Microsoft Device Manager. Contact Previsé if problem persists.
#3 - Emulation channel has been Terminated!	FATAL error. Power down and reboot both computers
#4 - Fatal Error! #6 - Error opening Kernel-Mode Driver!! #10 - Fatal Error! #14 - DLL failed to open lower ASC system DLL!	Required SCSI components were found, but failed to function. Check seating of SCSI adapter at target. Verify cable connected. Attempt reboot. Check installation. If persist contact Previsé.
#5 - NO Target established - possible input data error	Check syntax for SCSI CIU in default.INI file
#7 - SendData buffer overflow! Check condition returned. #8 - Null SendData buffer! #9 - ZERO-size SendData buffer! #11 - Bad VProc parameter #12 - Bad S_CLIENT_CDB_REPLY parameter! #13 - DLL failed to set trace output! #17 - no SCSI channel structure established! #19 - NULL buffer ptr for Data Out transfer #20 - NULL buffer ptr for Data In transfer #21 - NULL event handle(s) for post-DataOut #22. #27. #44	Non repairable at client site Contact Previsé
#16 - DLL thread failure!	Likely resulted from timeout of some thread. Reboot simulation host computer and restart simulator and advise Previsé that this error occurred.
#18 - Access violation exception - DLL level!	Reboot simulation host computer and restart simulator and advise Previsé that this error occurred.
Issues Related to Specific Function Code	
Optimal sequencing has been created for module x:x.x.	FC82 spec S15 called for optimal execution sequencing
XX block not supported yet	Describes Function Code not supported within current version of simulator. Contact supplier to have support for this FC added
RAM and other resource issues	
Insufficient Available Physical Memory (less than 50 Mbytes):	This message is issued every second when the amount of free physical memory is less than 50 Mbytes. The amount of used physical memory is close to the maximum available. You must add physical memory to this computer (i.e. add more RAM) or reduce the number of applications running on the computer.

Event Log and Debug Window Messages	
Message	Interpretation
DcsIOManager Issues	
Has been started. Update Rate: <Rate> Update Delay:<Delay> Definition <File Name> has been loaded Connection to BaileyDCSSimulator successful.	Normal DcsIOManager Startup information messages
Has stopped - by BaileyDCSSimulator Has stopped - BaileyDCSSimulator is not running. Has stopped - No IO defined Has stopped - Failed to connect to BaileyDCSSimulator. Has stopped - API is not enabled on BaileyDcsSimulator key.	DcsIOManager Shutdown information messages
Can not connect to OPC Server: <OPC Server> Can not add OPC Tag: <TagName> (Server: <OPC Server>) OPC Server:<OPC Server> disconnected	OPC Server connection issues. Verify addressing
Can not write value to OPC Tag: <TagName> (Server: <OPC Server>)	Verify OPC server tag address
Failed to open <File Name> Can not open file <File Name>	Verify file name, path, and file type
Unable to parse the connection(From-<addr>To-<addr>),	Verify FROM and TO address
Write To Tag has been already defined(<address>)	Eliminate duplicate
Couldn't open database--Exception: <Description>	Verify file to confirm not corrupt
Couldn't open IOTable in MDB file: <Description>	Verify IOTable name
Error reading columns in IOTable in MDB file: <File Name> Error reading IOTable in MDB file: <File Name>	Verify columns in IOTable against sample provided
Error reading columns in BaileyWrapTable in MDB: <File> Error reading BaileyWrapTable in MDB file: <File Name>	Verify columns in BaileyWrapTable against sample provided
Read Module - <address> Block-<address> Error - <Desc> Read Module - <address> Error-<Description> Write Module - <address> Block-<address> Error - <Desc> Write Module - <address> Error - <Description>	Specific addressing issues in IOTable or BaileyWrapTable
Other Issues	
Set Throttle - 1.0. Set Throttle - 1.9.	Change throttle setting
Time master Acc:12 Loop:5 Pcu:247 Tso:FFFECCB9E400 has been created.. CIU(5:1:2) Time sync rcvd: ACC(10>10) L=5 P=2 .	Time synchronization messages
Loop Back Mode	Loop Back mode ENABLED in DEFAULT.INI
Error Adding Module Error Adding CIU Error Adding Module Error Adding IO Module	Verify address and license limits. Contact supplier of problem persists
Block count: XX	Information message
Error creating Reply event. Error creating TCP event. Can't create overlapped event Can't create overlapped event	Internal problem – contact supplier

Appendix D – Process Simulator Interface (API)

This appendix provides a summary of DCOM API methods supported by the Bailey DCS Simulator, and intended for use to connect an external Process Simulation host. This interface provides the functions to support:

- Pause, resume, restart of Bailey DCS Simulator
- Throttle from 0.1 X to 10.0 Real Time
- Save MFP controller CFG files
- Read/write date and time
- Read/Write arrays of block data
- Save, restore and manage process state files (Initial Conditions)
- Log and replay operator commands

A summary of the methods available at this interface follows. This API is further described within The Previs Technical Manual, Bailey DCS Simulator API. Contact Previs if you require further information.

Method	Purpose
Simulator Control Methods	
Restart	Restart all modules currently configured into the DCS (via the <i>default.INI</i> file).
Pause	Pause all the modules currently configured into the DCS (via the <i>default.INI</i> file).
Continue	Continue (Resume) execution of paused modules.
Throttle	Change speed of simulation with respect to real time from 0.1 X to 10 X real time.
SaveCFGFiles	Save controller module binary CFG file overwriting CFG files already present, for any controller module CFG file configured into the DCS (via the <i>default.INI</i> file).
Date Time Methods	
DateTime	Read or write system date and time for computer hosting Bailey DCS Simulator
Process Data Read & Write Methods	
ReadBlockArray	Read an array of block values.
WriteBlockArray	Write an array of block values.
Simulator Process State File Methods	
DcsState	Provides functions to: <ul style="list-style-type: none"> ▪ Save DCS process state to a state file ▪ Restore a DCS state file ▪ Retrieve a list of all stored DCS state files ▪ Delete a specific DCS state file
Operator Command Log and Replay Methods	
CommandLog	Supports logging of operator commands via operator console including: <ul style="list-style-type: none"> ▪ Start command log. Writes incoming operator commands to retrieval buffer. ▪ Get buffer of time tagged operator commands logged to buffer. ▪ Reset (empty) operator command log buffer. ▪ Stop operator command logging. ▪ Replay single operator command

Appendix E – Support for Specific Connected Products

This appendix identifies the software products that are known to successfully connect to the Bailey DCS Simulator via the emulated CIU interface.

This is NOT to say that other products will not connect successfully, but rather the connection has not yet been tested or verified. Previs will add additional equipment to this list as soon as connection support is verified.

NOTE: If you successfully connect a software product not listed here, whether an ABB product or any third party product, please let us know. If you have problems connecting any software product to our emulated CIU please tell us so that we may resolve any connection problems.

Product Connection Support		
Product	Supported ?	Note
Engineering Workstation Software		
CADEWS	Supported.	Connect to CIU type SER via RS-232 only.
WinCAD	Supported.	Connect to CIU type ICT03 via RS-232 only.
Composer v4.0	Supported.	Use the following line in Default.INI file <i>CIUCHANNEL,ICT03:E0,<L>,<P>,<port>:<BaudRate></i>
Composer v5.0	Supported.	Use the following line in Default.INI file <i>CIUCHANNEL,ICT03:E0,<L>,<P>,<port>:<BaudRate></i>
WindowView	Supported.	Connect to CIU type ICT03 to gather trend values.
GMCL DBDOC		Not confirmed but should work via ICT03 serial RS-232
Operator Console Software		
OPsCon	Supported	Connect via any of: <ul style="list-style-type: none"> ▪ CIU type SER via RS-232 ▪ CIU type ICT03 via RS-232 or SCSI ▪ TELNET CIU via TCP/IP
Conductor NT (Ver 3.0 and 4.0)	Supported	Connect to CIU type ICT03 via RS-232 or SCSI
OIS4x	Supported	Connect to CIU type ICT03 via SCSI. Contact Previs for SCSI installation instructions and equipment.
Operate ^{IT} Process Portal A	Supported	Connect to CIU type ICT03 via RS-232 or SCSI. Contact Previs for SCSI installation instructions and equipment.
Operate ^{IT} Process Portal B	Supported	Connect to CIU type ICT03 via RS-232 or SCSI. Contact Previs for SCSI installation instructions and equipment.
PGP (Power Generation Portal)	Supported	Use the following line in Default.INI file <i>CIUCHANNEL,ICT03:E0,<L>,<P>,<port>:<BaudRate></i>
Wonderware InTouch	Supported	Connect via Wonderware driver for Bailey DCS to: <ul style="list-style-type: none"> ▪ CIU type SER via RS-232 ▪ CIU type ICT03 via RS-232 (SCSI not confirmed)
DeltaV Connect	Supported	Connect via CIU type ICT03 via RS-232 or SCSI
CiTech HMI	Supported	Connect via CiTech driver for Bailey DCS using CIU type SER via RS-232. We are aware that CiTech is making substantial enhancements to their driver, and newer versions may require connection via CIU type ICT03.
Driver and OPC Server		
SEMAPI	Supported	Connect via CIU type ICT03 via RS-232 or SCSI
Previs OPC Server	Supported	Connect via any of: <ul style="list-style-type: none"> ▪ CIU type SER via RS-232 ▪ CIU type ICT03 via RS-232 or SCSI ▪ TELNET CIU via TCP/IP
Rovisys OPC server	Supported	Connect via CIU type ICT03 via RS-232 or SCSI

Appendix F – Auto-Create CFG File

The *CFGAutoCreate.EXE* function, located in the Bailey DCS Simulator Program Files directory, can be used to automatically create a set of CFG files to match the configuration within the tag database CFG file that is output from the Previs Bailey DCS OPC Server and driver (CIUDRV.EXE).

NOTE: *CFGAutoCreate.exe* ONLY works with the unmodified format of the CSV file as provided by the Previs OPC Server and driver for Bailey DCS.

Function

CFGAutoCreate will:

- Open and read the CSV file and generate a *CSV File Format Error* message to the screen if invalid CSV file format.
- Create one module (i.e. one CFG file) for each module address represented in the CSV file. These CFG files are named according to the convention:

<LL><PP><MM>.cfg where:

LL = Loop address (positive integer, 1 or 2 digits, leading zeroes suppressed)

PP = PCU address (positive integer, 2 digits)

MM = Module address (positive integer, 2 digits)

- Create one block for each tag in the CSV file, according to the following table:
 - Type ANALOG generates FC30
 - Type DIGITAL generates FC45
 - Type STATION generates FC80
 - Type RMSC generates FC68
 - Type RCM generates FC62
 - Type DD generates FC123
 - Type MSDD generates FC129
 - Type RMCB generates FC136
 - Type TEXT generates FC151
 - Type DAANG generates FC177

Block Specifications

When the blocks are created, all blocks are provided default specification values (per Bailey DCS documentation) except the following:

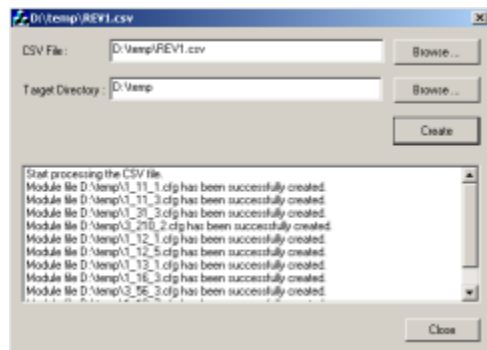
MSDD

S7 = 0
 S8 = 1
 S9 = 10
 S10 = 100
 S11 = 111
 S12 = 110
 S13 = 100
 S19 = 23
 S20 = 13
 S21 = 12
 S23 = 0

RMCB

S11 = 10
 S12 = 5
 S13 = 5

User Interface



Select the CSV file you wish to use as a source for CFG generation and the target directory to store the CFG files.

CFGAutoCreate displays messages about the created modules in its log window. If **CFGAutoCreate** finds an error during processing the CSV file, it displays a message in the log window:

Can not process the CSV file. Line – N

Where: N equals the line number which can not be processed. If line number equals 0 it's likely because the CSV file has the wrong format (i.e. not created by Previs OPC Server)

Appendix G – Details of Loop Back Mode

In Loopback Mode, the Bailey DCS Simulator is designed to provide signals for all console block types. For some block types these signals are based upon a general purpose Analog Signal and a general purpose Digital Signal.

General Purpose Analog Signal

The general purpose Analog Signal is used to provide an analog signal where required (e.g. FC30 input S1). This signal is a sine wave which:

- Is phased for each block address using the actual block address as a phase source
- Has a period of TA seconds as determined by the LOOPBACK record in the Default.INI file.

Base signals are used in the function codes to calculate inputs based on spec values of the particular block.

General Purpose Digital Signal

All digital input signals are generated on each interval defined by TD, and faced out by the time interval calculated as TD/address.

ANALOG (FC30)

Input value (S1) is calculated as:

$$\text{Input} = S3 + S4/2. + S4/2.*(\text{General Purpose Analog Signal})$$

DIGITAL (FC45)

Input value (S1) switches from 0 to 1 using time interval defined by TD (see *default.ini*).

RCM (FC62) simulation.

Loop back mode is the same as normal FC62 operation, though behavior is controlled by the choice of specifications selected.

RMSC (FC68) simulation.

Loop back mode is the same as normal FC68 operation, though behavior is controlled by the choice of specifications selected.

STATION (FC80) simulation.

To provide the ability to control the STATION block in Loop Back Mode the following specification values are set:

- S2 = 5
- S4 = 5

- S5 = 0
- S6 = 5
- S17 = 0
- S18 = 0
- S19 = 0
- S20 = 0
- S21 = 0
- S22 = 0
- S23 = 0
- S25 = 0
- S26 = 0
- S27 = 0
- S28 = 0
- S29 = 0
- S30 = 0

Input value (S1) is calculated as:

$$\text{InputS1} = \text{S11} + \text{S10}/2. + \text{S10}/2.* \text{ (General Purpose Analog Signal)}$$

Input value (S3) is calculated as:

$$\text{InputS3} = \text{S11} + \text{S10}/2. + \text{S10}/2.* \text{ (General Purpose Analog Signal)}$$

DD (FC123) simulation.

To provide the ability to control the DD block in Loop Back Mode the following specification values are set:

- S1 = 0
- S2 = 0
- S3 = 0
- S5 = 1
- S6 = 0

MSDD (FC129) simulation.

To provide the ability to control the MSDD block in Loop Back Mode the following specification values are set:

- S3 = 0
- S4 = 0
- S5 = 0
- S6 = 1
- S26 = 0

RMCB (FC136) simulation.

To provide the ability to control the RMCB block, the following specification values are set in Loop Back Mode:

- S1 = 0
- S2 = 0
- S3 = 1
- S4 = 1
- S5 = 1
- S6 = 1

- S7 = address
- S8 = address
- S9 = 1
- S10 = 1

TEXT (FC151) simulation.

In Loop Back Mode following specs values are set:

- S5= 1
- S6= 1
- S7= 0
- S8= 2
- S9= 2
- S10= 0
- S11= 3
- S12= 3
- S13= 1

The message number, color and blink status change periodically with interval TD (set via Default.INI) according to the values of specifications S5 to S13.

DAANG (FC177)

In Loop Back Mode the following specification values are set:

- S1 = 100
- S2 = 0
- S3 = 0
- S4 = 9.2e18
- S5 = -9.2e18
- S6 = 0
- S7 = 5
- S8 = 5
- S9 = 5
- S10 = 5
- S11 = 2
- S12 = 5
- S13 = 0
- S14 = 1
- S15 = 0
- S16 = 0
- S17 = 5
- S18 = 5
- S19 = 5
- S20 = 0
- S21 = 2
- S22 = 0
- S23 = 0
- S24 = 9.2e18
- S25 = -9.2e18
- S26 = 0
- S27 = 0

- S28 = 2
- S29 = 9.2e18
- S30 = 2
- S31 = 100
- S32 = 9.2e18
- S33 = 0
- S34 = 2

Input value (S10) is calculated as:

$$\text{InputS10} = S3 + (S1-S3)/2. + (S1-S3)/2.*\text{sSin}$$

Input value (S12) is calculated as:

$$\text{InputS12} = S3 + (S1-S3)/2. + (S1-S3)/2.*\text{sCos}$$

Appendix H – MCP02/ICT03 DIP switch settings

This Appendix provides DIP switch settings for Bailey CIU modules MCP02 and ICT03 as an aid to determining the SCSI Device ID for setting up emulated ICT03-SCSI channel.

MCP02 DIP Switch Settings

IIMCP02 DIP Switch Description			
Switch	Position	Description	Switch Settings / Range
1	1-4	Port A baud rate	1111 = 19200 , 0111 = 9600
	5-8	Port B baud rate	1111 = 19200, 0111 = 9600
2	1	MLM handshake timeout	0 = Enabled
			1 = Disabled
	2	MLM Diagnostic mode	0 = Disabled
			1 = Enabled
	3	Diagnostic utilities mode	0 = Disabled
			1 = Enabled
4	Hardware diagnostics	0 = Disabled	
		1 = Enabled	
5-8	Not used	-	
3	1	SCSI port	0 = Disabled
			1 = Enabled
	2-4	SCSI address	100 = 4
	5	SCSI parity check enabled	0 = Disabled
1 = Enabled			
6-8	Not used	-	
4	1	ROM checksum	0 = Enabled
			1 = Disabled
	2-3	Serial Port Parity settings	00 = 8 d, 1s, no parity
			01 = 8 d, 1s, even parity
			10 = 8 d, 1s, odd parity
			11 = 8 d, 1s, no parity
	4	Serial Port B Mode	0 = NIU command mode
			1 = NIU utility mode
	5	Modem Password protection	0 = Disabled
			1 = enabled
	6	Port addressing mode	0 = Disabled
			1 = enabled
	7	Command checksum	0 = Disabled
			1 = Enabled
	8	Security ENABLE MCP02 only	1 = Security Enabled
			0 = Security Disabled
NOTE: 0 = CLOSED or ON, 1 = OPEN or OFF. Bold areas indicate factory default settings.			

ICT03 DIP switch settings

INICT03 DIP Switch Description			
Switch	Position	Description	Switch Settings / Range
1	1-4	Port A baud rate	1111 = 19200 , 0111 = 9600
	5-8	Port B baud rate	1111 = 19200, 0111 = 9600
2	1	MLM handshake timeout	0 = Enabled 1 = Disabled
	2	MLM Diagnostic mode	0 = Disabled 1 = Enabled
	3	Diagnostic utilities mode	0 = Disabled 1 = Enabled
	4	Hardware diagnostics	0 = Disabled 1 = Enabled
	5-7	Not used	-
3	1	SCSI port	0 = Disabled 1 = Enabled
	2-4	SCSI address	100 = 4
	5	SCSI parity check enabled	0 = Disabled 1 = Enabled
	6-8	Not used	-
4	1	ROM checksum	0 = Enabled 1 = Disabled
	2-3	Serial Port Parity settings	00 = 8 d, 1s, no parity 01 = 8 d, 1s, even parity 10 = 8 d, 1s, odd parity 11 = 8 d, 1s, no parity
	4	Serial Port B Mode	0 = NIU command mode 1 = NIU utility mode
	5	Modem Password protection	0 = Disabled 1 = enabled
	6	Port addressing mode	0 = Disabled 1 = enabled
	7	Command checksum	0 = Disabled 1 = Enabled
	8	Security ENABLE MCP02 only	1 = Security Enabled 0 = Security Disabled
NOTE: 0 = CLOSED or ON, 1 = OPEN or OFF. Bold areas indicate default settings.			

Appendix I – Format of Points Table Report

This appendix describes the format and contents of the Points Table report as exported from the simulator at the Debug Tab Points Table function.

Purpose of this Report

The purpose of the points table report is:

- To reveal exactly what points have been established by a connected operator console.
- To reveal details of each connected point including current data values.
- To identify any console points that do not connect to existing function blocks within the controller logic. (e.g. L:P:M:B address of console tag does not exist within logic)
- To identify any console points that have a data type mismatch with the block type that they are connected to. (e.g. ANALOG point type connected to DIGITAL block type).
- To provide statistics of exception traffic received from emulated controllers and sent to console.

Sample Report

Index	Type	loop	PCU	mod	block	match	est	conn	FromModu	ToConsole	Command	Data Q	CV..			
6	5	1	1	5	30	2	1	0	4	2	0 00h			9		
36	7	1	1	5	45	2	1	0	4	2	0 00h					
37	17	1	1	5	50	2	1	0	4	2	0 01h	0.015625		9	9.20E+18	
38	15	1	1	5	1129	2	1	0	4	2	0 00h					
39	15	1	1	5	1136	2	1	0	4	2	0 00h					
40	5	1	1	5	1151	2	1	0	4	2	0 00h	0.000137				
41	15	1	1	5	1062	2	1	0	4	2	0 00h					
42	19	1	1	5	1068	2	1	0	4	2	0 00h	0				
43	17	1	1	5	1080	2	1	0	4	2	0 00h	0		0		0
44	15	1	1	5	1123	2	1	0	4	2	0 00h					

Format of Points Table Report

The fields within the point table report are:

- **Index** – the integer point index of the points table entry.
- **pointType** – The point type of the table entry, as selected from the accompanying Point Type table
- **Loop, PCU, Mod, Block** – L:P:M:B address for this point

- **Match** – Database match indicator as follows:
 - 0 – No block at L:P:M:B address (address mismatch)
 - 1 – Block exists at point L:P:M:B address but does not support Point Type from console (type mismatch)
 - 2 – Function block addresses by this point exists and supports selected Point Type (no apparent problem)
- **Established** – Point established (1) in Point Table or not (0)
- **Connected** – Point connected (1) and currently subscribing data to the console or not (0).
- **ExFromModule** – Count of the number of exceptions received from the emulated module for this point since last counter reset.
- **ExToConsole** – Count of the number of exceptions sent to the console for this point since last counter reset.
- **ExFromConsole** – Count of the number of console commands received for this Point since last counter reset.
- **Data Q CV..** – Current process data for this point as follows:
 - For point types 0, 6, 7, 11, 13, 14, 15 and 23 presents variable number of bytes in hex. If no bytes available presents "Point Size Undefined"
 - For point types 1, 2, 3, 4, 5, 8, 9, 10, 12 and 19 presents first byte in hex (point status byte) and the process value in floating point form.
 - For point type 17 presents first byte in hex (status byte) and the process variable, set point and control output in floating point form.
 - For point type 29 presents fist byte in hex (status byte), second, third and forth bytes in hex (extended status) and the output value, next higher limit and next lower limit in floating point form.
 - For point type 21 presents fist byte in hex (point status byte) and the process value in floating point form
 - For point type 28 presents no data.

Point Type table			
Type	Meaning	Type	Meaning
QUALITY MANAGEMENT		DIGITAL	
-1	Undefined	7	DIGITAL Read
0	Quality	15	RCM Read
MODULE STATUS		ANALOG	
14	MODSTAT Read	5	REAL3 Analog Read
23	EXT MODSTAT Read	21	REAL4 Analog Read
CONSOLE STATION READ / WRITE		19	RMSC Read
1	STATION PV Read	29	DAANG Read
2	STATION SP Read	CONSOLE OUTPUT REPORTS	
3	STATION CO Read	12	REAL3 Analog Report
4	STATION RI Read	13	DIGITAL Report
6	STATION STATUS Read	16	RCM Report
8	STATION SP Write	18	STATION Single Report
9	STATION CO Write	20	RMSC Report
10	STATION RI Write	22	REAL4 Analog Report
11	STATION Mode Write	TREND	
17 ⁶	STATION Single Read	28	Trend Point

⁶ Type 17 is also used for DAANG data

Appendix J – Setup RTTgSCSI.ini for SCSI channel

When a SCSI channel is defined using the RTST line in default.ini, there is an additional INI file, named RTTgSCSI.ini that contains additional setup parameters. You will usually not need to change this INI file. But if you do, this appendix describes the contents of this file.

A sample RTTgSCSI.ini is provided here. You may locate this file in the .\BaileyDCSSimulator\hwininstall\directory.

The parameters in this file are as follows:

```
[Startup]
ShowDialog=TRUE
[Logging]
Enabled=FALSE
[Trace]
Enabled=FALSE
[SCSI]
Channel=0
TargetId=4
Lun=0
BusyFlag=FALSE
SendDisco=FALSE
ReceiveDisco=TRUE
ResponseDelay=2
```

- [Startup]
 - ShowDialog=TRUE - TRUE presents start dialog for debug use; FALSE shows build date if run manually
- [Logging]
 - Enabled=FALSE (do not change this)
- [Trace]
 - Enabled=FALSE (do not change this)
- [SCSI]
 - Channel=0 (no need to change this, this is controlled by simulator)
 - TargetId=4 (no need to change this, this is controlled by simulator)
 - Lun=0 (no need to change this, this is controlled by simulator)
 - BusyFlag=FALSE (TRUE allows SCSI command Test-Unit-Ready to poll CIU-Busy status – change only at Previs request)
 - SendDisco=FALSE (TRUE allows Target to disconnect from Host if slow processing for SCSI Send command – change only at Previs request)
 - ReceiveDisco=TRUE (TRUE allows Target to disconnect from Host if slow processing for SCSI Send command – change only at Previs request)
 - Response Delay (in milliseconds) – provides a short delay before command response to allow VAX (or other console) SCSI hardware to get ready for response. This value if 2 milliseconds was required with VAX)

Appendix K – Set Up DCSIOManager

The *DCSIOManager* application, installed at the `..\Program Files\Bailey DCS Simulator\..` directory, is a data router. The specific functions performed are:

- Read data from defined block address in Bailey DCS Simulator, and write it to another block address. This supports the connection of plant process simulation logic in one controller to process control logic in another.
- Read data from API and write via OPC Data Access 2.0 Client interface, to any OPC Data Access server.
- Read data via OPC Data Access 2.0 Client interface, from OPC Data Access server and write it via API to Bailey DCS Simulator.

The DCSIOManager functions are controlled via an MDB Database file, which provides the data source and destination tag database.

Starting the DCSIOManager

The *DCSIOManager* application is started automatically by the Bailey DCS Simulator application if, and only if, the IOMANAGER line is included within the *default.ini* file.

Configuring the DCSIOManager

The DcsiIOManager.INI file, located at the `..\Program Files\Bailey DCS Simulator\..` directory, is used to configure the DCSIOManager. Sample contents for this file are illustrated here.

```
[MAIN]
RATE=250
DELAY = 5000
[DEFINITION]
io1at5rev2.mdb
```

The settings in this file are:

- Section [MAIN]
 - RATE defines the read/write frequency in milliseconds.
 - DELAY defines a startup delay in milliseconds.
- Section [DEFINITION]
 - Contains only the filename, located at the `..\Program Files\Bailey DCS Simulator\..` directory, that contains the tag database to configure the routing functions of the DCSIOManager. This file MUST be a Microsoft Access MDB file.

The Tag Database MDB File

The Microsoft Access MDB file that will contain the tag database for DcsIOManager may contain 1 or 2 tables. You may use one table, or both, but no other tables will be used. These names of these table MUST be:

- **IOTable** – This table supports:
 - Routing signals from Source address within the Bailey DCS Simulator to a Destination address within the Bailey DCS Simulator.
 - Routing signals from Source address at OPC Data Access Server to Destination address within the Bailey DCS Simulator.
 - Routing signals from Source address within the Bailey DCS Simulator to Destination address at OPC Data Access Server.
- **BaileyWrapTable** – This table supports ONLY routing signals from a Source address within the Bailey DCS Simulator to a destination address within the Bailey DCS Simulator. The addressing method used within BaileyWrapTable is easier to use than that within IOTable.

IOTable Record Format

The contents of the IOTable within the MDB database file are illustrated here.

WRITEFILE	READFILE	COMMENT	READADDRESS	WRITEADDRESS	OPERATION	WRITESOURCE	READSOURCE
BMS I/O		MILL H 10% IGNITOR	[AT5SIM]N10:021/15	5:53:20:12287	3	BaileyDcsSimulator	RSLinX OPC Server
BMS I/O		MILL C 4% IGNITOR	[AT5SIM]N10:021/04	5:53:20:12272	3	BaileyDcsSimulator	RSLinX OPC Server
5530418	55320C6	MFT RELAY STATUS TRIP#	5:53:20:8329	5:53:4:592	3		
5530418	55320C6	FW BYPASS VLV OPENED#	5:53:20:8327	5:53:4:590	3		
	BMS I/O	HIGH FURN PRESS TRIP A	5:53:20:16134	[AT5SIM]N10:018/01	3	RSLinX OPC Server	BaileyDcsSimulator
	BMS I/O	AIR DAMPERS AT PURGE	5:53:20:12327	[AT5SIM]N10:018/00	3	RSLinX OPC Server	BaileyDcsSimulator

In this example there are 6 records, with functions as follows:

- The 1st and 2nd records read data from an RsLinX OPC server and write it to the Bailey DCS Simulator.
- The 3rd and 4th records read data from one block address in the Bailey DCS Simulator and write to another block address within the simulator.
- The 5th and 6th records read data from the Bailey DCS Simulator and write it to the an RsLinX OPC server.

The fields in this file are defined as follows:

- **WRITEFILE** – Free form text comment about write source
- **READFILE** – Free form text comment about read source

- COMMENT – Free for comment field, used for tag name
- READADDRESS - The address within READSOURCE that this record reads from.
- WRITEADDRESS - The address within WRITESOURCE that this record writes to.
- OPERATION – Coded operation, with alternatives as follows:
 - 1 – Write Field IO
 - 2 – Write Program IO
 - 3 – Write Forced value
 - 4 – Clear previously forced value
 - 5 – Write Specification
 - NOTE: Use OPERATION CODE 3 for ALL field IO write operations except the following:
 - Use operation code 1 to write count rate, in counts per second, to FC104 output N. This permits Fc104 to maintain correct totalization function..
- WRITESOURCE – The address string for the device to be written to. Use “BaileyDCSSimulator” to write to Bailey DCS Simulator.
- READSOURCE - The address string for the device to be read from. Use “BaileyDCSSimulator” to write to Bailey DCS Simulator.

BaileyWrapTable Record Format

The required fields within BaileyWrapTable are defined as follows:

- **Description** (text) – This is a textual description of this record.
- **S_L** (integer) – Source Loop Address
- **S_P** (integer) – Source PCU Address
- **S_M** (integer) – Source Module Address
- **S_B** (integer) – Source Block Number
- **S_S** (integer) – Source Specification number, as follows:
 - If **S_S = 0** or **S_S = NULL** (empty) , then DCSIOManager will perform DIRECT READ from the Block address provided.
 - If **S_S = valid specification number (i.e. 3, 4, 5)** , then DCSIOManager will perform an INDIRECT READ of the upstream value at the address provided by the specification.
- **D_L** (integer) – Destination Loop Number

- **D_P** (integer) – destination PCU address
- **D_M** (integer) – Destination Module Address
- **D_B** (integer) – Destination block number
- **D_O** (integer) – Destination operation, selected from:
 - 1 – Write Field IO
 - 3 – Write Forced value
 - NOTE: Use OPERATION CODE 3 for ALL field IO write operations except the following:
 - Use operation code 1 to write count rate, in counts per second, to FC104 output N. This permits FC104 to maintain correct totalization function..
 - NOTE: If the D_O operation is neither 1 or 3, then the value 3 is assumed. If the D_O column is not present, the value 3 is assumed for all destination write operations.
- **Default** (text) – This is a default numeric value (i.e. 23.5, 1) that will be written to the destination address, if no source address is provided.

Other fields may be included as well, but the above fields are required.