



































## OPC90 Server

For Bailey

NETWORK 90<sup>®</sup> / INFI 90<sup>®</sup> / Symphony Plus

NETWORK 90 and INFI 90 are registered trademarks of ABB (Formerly Bailey Controls Company).

<b>1</b>	<b>INTRODUCTION .....</b>	<b>6</b>
<b>2</b>	<b>FUNCTIONAL DESCRIPTION.....</b>	<b>7</b>
<b>3</b>	<b>INSTALLATION .....</b>	<b>8</b>
<b>4</b>	<b>BAILEY COMMUNICATION OVERVIEW.....</b>	<b>9</b>
<b>5</b>	<b>OPC90 SERVER APPLICATION MENU ITEMS.....</b>	<b>14</b>
5.1	FILE.....	15
5.1.1	New.....	15
5.1.2	Open .....	15
5.1.3	Save.....	16
5.1.4	Save As.....	16
5.1.5	Auto Save.....	16
5.1.6	Import CSV.....	16
5.1.7	Reimport CSV.....	17
5.1.8	Export CSV.....	17
5.1.9	Reexport CSV.....	17
5.2	ADD.....	17
5.2.1	Device.....	18
5.2.2	Group.....	19
5.2.3	Block.....	19
5.2.3.1	Alarm Hold .....	21
5.3	EDIT .....	21
5.3.1	Deletion .....	21
5.3.2	Com Ports.....	22
5.3.3	TCP Ports .....	23
5.3.3.1	Setting Up Lantronix UDS 2100 .....	24
5.3.4	Engineering Units.....	27
5.3.5	Text Messages.....	28
5.3.6	Red Tag Users .....	29
5.3.7	SOE Point Names .....	31
5.3.8	Properties .....	32
5.3.9	Priority .....	32
5.3.10	Configuration Shadowing Between Redundant Servers .....	32
5.3.10.1	Redundant Same Address CIUs.....	35
5.3.10.2	OPC Client Access of Redundant Servers .....	36
5.3.11	Error Detection.....	36
5.3.12	Tree View Settings .....	39
5.4	VIEW .....	40
5.4.1	View Monitor .....	41
5.4.2	View Status .....	42
5.4.3	View Start In Runtime.....	43
5.4.4	View Run As Service.....	43
5.4.5	View Lockout OPC Access.....	44
5.4.6	View Refresh.....	44
5.4.7	View Inactivity Time .....	44
5.5	UTILITIES .....	45
5.5.1	Service Manager.....	45
5.5.2	DCOM Configuration.....	45
5.5.3	View My Logs .....	45
5.6	BLOCK FACEPLATES.....	45
5.6.1	Faceplate Panels .....	46
5.6.1.1	Display .....	47
5.6.1.2	Save Panel.....	47

5.6.1.3	Open Panel.....	48
5.6.1.4	Close Panel.....	48
5.6.1.5	Minimize All.....	48
5.6.1.6	Show All.....	48
5.7	HELP.....	48
5.7.1	Block Finder.....	48
<b>6</b>	<b>GROUPS.....</b>	<b>51</b>
<b>7</b>	<b>BLOCKS.....</b>	<b>52</b>
7.1	 ANALOG INPUT LOOP (AIL).....	55
7.2	 ANALOG OUTPUT LOOP (AOL).....	57
7.3	 BLOCK DATA (BLK).....	59
7.3.1	Reading a Bailey Block.....	61
7.3.2	Tuning a Bailey Block.....	61
7.3.3	Reading the Entire Bailey Configuration.....	61
7.3.4	Changing Bailey Controller Module Modes.....	62
7.3.5	Writing Bailey Controller Blocks.....	63
7.3.6	Modifying Bailey Controller Blocks.....	63
7.3.7	Deleting Bailey Controller Blocks.....	64
7.3.8	Configuration Command Processing.....	64
7.3.9	BLK Configuration Faceplate.....	69
7.4	 DATA ACQUISITION ANALOG (DAANG).....	73
7.5	 DATA ACQUISITION DIGITAL (DADIG).....	75
7.6	 DATA BLOCK (DATA).....	77
7.7	 DEVICE DRIVER (DD).....	78
7.8	 DEVICE DEFINITION BLOCK (DEVICE).....	79
7.8.1	SCSI Communication.....	90
7.8.2	TCP Communication.....	93
7.8.3	Device Communication Faceplate.....	95
7.9	 DIGITAL INPUT LOOP (DIL).....	98
7.10	 DIGITAL OUTPUT LOOP (DOL).....	99
7.11	 HARMONY ANALOG INPUT (HAI).....	101
7.12	 HARMONY ANALOG OUTPUT (HAO).....	102
7.13	 HARMONY DIGITAL INPUT (HDI).....	103
7.14	 HARMONY DIGITAL OUTPUT (HDO).....	104
7.15	 MODULE STATUS (MODSTAT).....	105
7.16	 MULTI-STATE DEVICE DRIVER (MSDD).....	107
7.17	 MULTIPLEX CIU (MUXCIU).....	109
7.17.1	MUXCIU Virtual Port API.....	112
7.18	 NETWORK INTERFACE SLAVE MONITOR (NISMON).....	114
7.19	 OUTPUT DEVICE DRIVER (ODD).....	122
7.20	 OUTPUT MULTI-STATE DEVICE DRIVER (OMSDD).....	124
7.21	 OUTPUT REMOTE CONTROL MEMORY (ORCM).....	127
7.22	 OUTPUT REMOTE MOTOR CONTROL (ORMC).....	129
7.23	 OUTPUT REMOTE MANUAL SET CONSTANT (ORMSC).....	131
7.24	 OUTPUT STATION CONTROL (OSTN).....	133
7.25	 PCU COMMUNICATION STATISTICS MONITOR (PCUMON).....	136
7.26	 POLL ANY BLOCK (POLL).....	144
7.27	 REMOTE CONTROL MEMORY (RCM).....	146
7.28	 REMOTE MOTOR CONTROL (RMC).....	148
7.29	 REMOTE MANUAL SET CONSTANT (RMSC).....	150
7.30	 SEQUENCE OF EVENTS (SOE).....	151
7.31	 SPECIFICATION BLOCK (SPEC).....	156
7.32	 STATION PID CONTROL (STN).....	158

7.32.1	Station PID Control (STN) Faceplate .....	161
7.32.2	Station PID Control (STN) Trend and Tune Faceplate .....	162
7.33	TEXT SELECTOR (TXT) .....	164
7.33	TEXT STRING (TEXTSTR) .....	165
<b>8</b>	<b>OPC STATUS .....</b>	<b>166</b>
<b>9</b>	<b>CSV FILE FORMATS .....</b>	<b>167</b>
<b>10</b>	<b>SETTING UP OPC CLIENT ACCESS OF OPC90 .....</b>	<b>172</b>
10.1	CONFIGURING DCOM ON THE LOCAL OPC90 PC .....	172
10.2	CONFIGURING DCOM ON THE REMOTE OPC CLIENT PC .....	179
10.3	VALIDATING OPC CLIENT ACCESS OF OPC90 .....	181
<b>11</b>	<b>USING OPC90 WITH IET800 .....</b>	<b>184</b>
11.1	IET800 SWITCH SETUP .....	184
11.2	IET800 PARAMETER SETUP .....	185
11.3	IET800 ETHERNET PORT SETUP .....	190
11.4	OPC90 ETHERNET COMMUNICATION SETUP .....	190
11.5	IET800 SERIAL PORT SETUP .....	193
11.6	OPC90 SERIAL COMMUNICATION SETUP .....	193
11.7	PARTS SUMMARY .....	195
<b>12</b>	<b>MIGRATING OPC90 TO A WINDOWS 64 BIT ENVIRONMENT .....</b>	<b>195</b>
<b>13</b>	<b>TROUBLESHOOTING HINTS .....</b>	<b>197</b>
13.1	SENTINEL.SYS CAUSES PC BLUESCREEN ON BOOTUP .....	198
13.2	VALIDATING STATE OF INDIVIDUAL OPC90 SERVER BLOCKS .....	198
13.3	SCSI PORT NOT VISIBLE IN DEVICE BLOCK PROPERTIES COMM PORT LIST BOX .....	199
13.4	SCSI COMMUNICATION NOT WORKING ON DELL PC .....	199
13.5	SCSI ADDRESSES DIFFERENT BETWEEN TWO SHADOWED PCs .....	199
13.6	NO COMMUNICATION .....	200
13.7	COMMUNICATION PORT IS REPORTED AS BAD OR CAN'T OPEN .....	201
13.8	APPEARS TO BE COMMUNICATING BUT NO DATA IS BEING RECEIVED .....	202
13.9	NOT ALL BLOCKS ARE RECEIVING DATA .....	202
13.10	RANDOM BLOCKS ARE RECEIVING BAD QUALITY INDICATION .....	203
13.11	OSI PI CANNOT READ VT_BOOL TAGS CORRECTLY .....	203
13.12	SPORADIC DATA TRANSFER OF EXPORT BLOCKS TO BAILEY .....	204
13.13	CANNOT EXPORT OR CONTROL DATA WITHIN BAILEY .....	204
13.14	CANNOT SEE ALL DOCUMENTED BLOCK ATTRIBUTES .....	205
13.15	OPC90 SERVER WILL NOT TIME SYNC BAILEY .....	205
13.16	OPC CLIENT DOES NOT SEE ANY OPC SERVERS WHEN BROWSING REMOTE PC .....	205
13.17	OPC CLIENT CONNECTS BUT DOES NOT RECEIVE DATA .....	208
13.18	VIEWING DATA IN MONITOR MODE DOES NOT WORK .....	208
13.19	WRITING RMSC BLOCKS CAUSE CONTROLLER OR NODE OFFLINE CONDITION .....	208
13.20	LAST SAVED DATABASE NOT AUTOMATICALLY RESTORED ON PROGRAM START .....	208
13.21	DATA UPDATES STOP AFTER 1 HOUR .....	208
13.22	DATA UPDATES STOP AFTER 8 HOURS .....	209
13.23	OPC90 RUNNING AS A SERVICE RANDOMLY STOPS OR CRASHES .....	210
13.24	CAN'T RUN AS A SERVICE UNDER WINDOWS .....	210
13.25	BLK AND SPEC BLOCKS DO NOT REPORT CORRECT SPECIFICATION NAMES .....	211
13.26	CIU RED LIGHTS WITH LEDs 1, 4, 5 AND 6 ON .....	211
13.27	OPC90 WILL ONLY RUN IN DEMO MODE OR REPORTS DEMO EXPIRED .....	211
13.28	FORCING A CIU RESTART .....	212
13.29	CAN'T ESTABLISH COMMUNICATION WITH IET800 .....	213
13.30	IET800 ETHERNET COMMUNICATION IS REALLY SLOW .....	213

13.31	BLOCKS REPORTING WAITING FOR SPECIFICATIONS.....	213
-------	--	-----

## 1 Introduction

The RoviSys Company has developed a tightly integrated interface between OPC clients and a Bailey DCS that promotes future plant upgrades using OPC. This interface is called the OPC90 Server. It allows easy and extremely user intuitive replacement of Bailey operator consoles with Windows based OPC client consoles. It also provides an open migration path from the Bailey control platform to OPC as plant time, resources and schedule permits.

The OPC90 Server enables seamless DCS access to the Bailey Controls Command Series, NETWORK 90, INFI 90, Harmony and Symphony Plus Distributed Control Systems. The server can communicate at the network level via the appropriate Computer Interface Unit (NCIU0X), Plant Loop to Computer Interface (INPCI0X), Infi-net to Computer Interface (SPIET800, INICI01, INICI03, INICI12, INICI13, IIMCP01/02) or, within a single Process Control Unit, via a Computer Interface Command (CIC01), Serial Port Module (NSPM01, IMSPM01) and Computer Port Module (CPM02/03). The bottom line is OPC90 can communicate with all interfaces available for NETWORK 90 / INFI 90 / Harmony / Symphony Plus and Command Series Distributed Control Systems. System integrity, functionality, and data throughput is maintained by utilizing standard exception reporting techniques. Bandwidth improvements are realized over existing Bailey use of these same hardware interfaces by implementing dual channel capability and redundant interfaces.

This document is intended for individuals who are familiar with the Bailey configuration principals in terms of the types and how to access exception report blocks. An understanding OPC is also required.

OPC90 Server is an OPC Data Access 1.0, 2.0 and 2.05 compliant server. This is achieved by using an industrial grade OPC toolkit to implement its OPC interfaces. The ongoing maintenance and certification of this toolkit is managed by a company that is a charter member of the OPC Foundation.

OPC90 Server supports DCOM. DCOM allows one PC running OPC90 to provide its data via a local area network to other PCs running OPC client software. Refer to the OPC90 Remote Access section of this document for instructions on how to setup DCOM.

Throughout this document the OPC90 Server will be referenced as OPC90. The Bailey system is referenced as *BLY* which is defined to include Command Series, NETWORK 90, INFI 90, Harmony and Symphony Plus systems.

## 2 Functional Description

OPC90 Server can run on the Windows XP / 2003 Server / 7 / 10 / 2008 / 2012 / 2016 Server operating systems. It's implemented as a collection of blocks that are specific to each type of exception report block found within a Bailey system. This collection of blocks is stored as an OPC90 Configuration. A special block called the Device block (DEVICE) is configured to setup the communication port(s) used to access the Bailey interface, define required update rates and report various statuses of the communication channels. All other OPC90 Server blocks are linked to a DEVICE that provides the necessary information it needs to manage communication with the Bailey system. The various other OPC90 Server blocks linked to the DEVICE contain the address within Bailey where the exception report block is located. The DEVICE uses this address and block type to determine how the point is established and managed within the Bailey interface. As data is received from the Bailey interface for each of the established points it is parsed and copied to the appropriate tags within the corresponding OPC90 Server block. This implementation has several advantages:

- Extremely intuitive to Bailey users.
- Developed with an industrial grade OPC toolkit manufactured by an OPC Foundation Charter Member company insures 100% compliance to the OPC 1.0, 2.0 and 2.05 specifications.
- Self validated using the OPC Foundation testing tools
- Point capacity is not limited by type of points utilized by the plant but only by the type of Bailey interface available (typically up to 30,000 tags).
- Alarm levels are set in real time by data received from the Bailey system (eliminates the need to maintain alarm level settings in two areas).
- Includes ability to tune Bailey control loops from OPC client applications.
- Supports redundant Bailey interfaces or redundant communication channels to a single Bailey interface.
- Supports the ability to easily import and export OPC90 function blocks to/from comma delimited (\*.csv) files. The CSV file can be easily created using Microsoft Excel and then simply imported into the OPC90 Server to create an OPC90 configuration.

### 3 Installation

The installation procedure utilizes the industry standard Install shield Setup program. Do the following procedure to install OPC90. This procedure assumes that the OPC90 Server is being installed on a Windows XP / 2003 Server / Windows 7 /10 / 2008 / 2012 / 2016 Server operating system.

- 1.) Log into a user account with administrator privilege.
- 2.) Insert the OPC90 Server installation CD into the CD drive.
- 3.) Run the setup program found in the root directory of the CD.
  - a. Right click on it and select Run as Administrator.
- 4.) Follow the instructions given by the Setup program.
- 5.) If using hardware licensing insert the RoviSys USB license key.
- 6.) If using software licensing run the RoviSys License Manager program and request a license. Until the license activation code is received setup a couple day demo period.
- 7.) You have completed installation of OPC90 Server!

After installation on Windows 7 right click on the C:\Program Files\OPC90 Server\OPC90Server.exe file, select properties and set compatibility mode to Windows XP. This will allow OPC90 to run as a service under Windows 7 and communicate with SCSI CIUs.

When upgrading, first uninstall using Windows add/remove programs and then run the above procedure. If running OPC90 as a service, make sure it is stopped before uninstalling.

Note that the demo version of OPC90 downloaded from the RoviSys website does not support software licensing. It must be uninstalled and the official version installed from the CD to accomplish licensing of the OPC90 software.



## 4 Bailey Communication Overview

Bailey utilizes a communication technique for data exchange called exception reporting. Exception reporting means the data is sent when it has changed significantly or a maximum time has expired since the last time it was sent. All exception reported data is made available to the rest of the system via a set of exception report related function blocks which are matched by an equivalent OPC90 Server block. Bailey has a series of RS232 serial based devices generically called Computer Interface Units that are utilized to receive this exception report data. Note that communication with the INICI03 interface can be either RS232 or SCSI. RoviSys is extremely knowledgeable on the use of the various Bailey interfaces and has implemented a single driver that can utilize the following types of interfaces:

Bailey Interface	Max. Number Bailey Blocks	Exception Reporting	Control Operation
NSPM01	500	No	No
IMSPM01	500	No	No
IMCPM02	500	No	No
IMCPM03	500	Yes	Yes
NCIC01	500	Yes	Yes
NCIU01	500	Yes	Yes
NCIU02	2,500	Yes	Yes
NCIU03	5,000	Yes	Yes
NCIU04	10,000	Yes	Yes
INPCI01	500	Yes	Yes
INPCI02	5,000	Yes	Yes
IIMCP01	10,000	Yes	Yes
IIMCP02	30,000	Yes	Yes
INICI01	10,000	Yes	Yes
INICI12 (see note 1)	10,000	Yes	Yes
INICI13 (see note 1)	30,000	Yes	Yes
INICI03 (see note 1)	30,000	Yes	Yes
SPIET800 (see note 2)	30,000	Yes	Yes

- 1.) When using these interfaces, the ABB Bailey semAPI software environment is not required. The INICI12, INICI13, and INICI03 do not support dual channel serial communication. When using serial communication to the INICI12, INICI13 and INICI03, make sure the cable is attached to the port labeled "terminal" on the ABB Bailey termination unit or termination module. See the sections "Add" and "Device Definition Block" for information regarding SCSI communication with the INICI03.

- a. INICI03 sites containing INICT03A firmware revision 'F' must upgrade to revision 'G' or later or downgrade to revision 'E'.
  - b. INICI13 is comprised of the INICT13A and INNIS01 modules. The default factory settings for the INICT13A SW4 are incorrect. The INICT13A SW4 Poles 6, 7, 8 must be open (1) when using SCSI with this module.
- 2.) OPC90 supports Ethernet or serial communication with IET800. IET800 firmware revision A7 or later is required for Ethernet communication. See "Using OPC90 with IET800" section for details on setting it up.

Throughput of the supported Bailey CIU devices is difficult to calculate empirically since the data is received and sent by exception and varies according to the total blocks in the database, data types of those blocks and what has changed significantly at any given time. The exception report rate per general data type for RS232 and SCSI interfaces presented in following table. The Ethernet style CIU is 2-3 times faster than SCSI.

<b>Data Type</b>	<b>RS232 @ 19.2k values / second</b>	<b>SCSI values / second</b>
Analog	260 / 161*	1080 / 1059*
Digital	310 / 260*	1230 / 1206*
Control Loop	106 / 91*	618 / 606*
<b>Average</b>	<b>225 / 170*</b>	<b>976 / 957*</b>

























\* Rate when the device block "Use DCS Timestamp" option is enabled.

A practical method for considering the use of serial versus SCSI is based on the total number of blocks in the database. So for example assume the total number of blocks is around 1000 and also assume 10% of those blocks are changing significantly. That would be a very active plant which would generate 100 exception reported values per second. For this example, performance of both the serial and SCSI based CIUs would practically be identical in terms of data updates per second since both can read 100 values within a second.

If the database was much larger, such as 5000 blocks and 10% changing significantly per second, serial versus SCSI takes on new meaning. The effective data load would be 500 values per second. It would take the serial interface 2.2 seconds to read 500 values whereas the SCSI interface could read them in 0.5 seconds.

From this analysis we see that from a practical standpoint the advantage of SCSI over serial becomes apparent only for large databases. The performance for databases of 2000 blocks or less can be considered only marginally better.

Note that a few of the above mentioned devices do not support exception reporting. Specifically the NSPM01, IMSPM01, IMCPM01 and IMCPM02. Data collection from these devices is basically limited to the OPC90 Server blocks that poll to get the block values. The names of these blocks are POLL, BLK and SPEC. The remaining interfaces support following OPC90 block type and associated Bailey exception reports types.

OPC90 Server Block Type	Bailey Block Name	F.C. Number
AIL 	Analog Output / Loop – AOL, AOLDB	30, 48, 70, 158
BLK 	Bailey Block Configuration	any function code
DAANG 	Data Acquisition Analog – DAANG	177
DADIG 	Data Acquisition Digital – DADIG	211
DD 	Device Driver – DD	123
DIL 	Digital Output / Loop – DOL, DOLDB	45, 67
HAI 	Harmony Analog Input	222
HAO 	Harmony Analog Output	223
HDI 	Harmony Digital Input	224
HDO 	Harmony Digital Output	225
MODSTAT 	Module Status	any module
MSDD 	Multi-State Device Driver – MSDD	129
MUXCIU 	Multiplexes a PC COM port to Bailey Interface for configuration related software (i.e. Bailey CADEWS, Composer, G.Michaels DBDOC, etc)	any function code
NISMON 	Provides detailed node and communication loop diagnostic information. Logs potential problems.	any node
PCUMON 	Monitors performance of a PCU node	any PCU node
POLL 	Poll Value - POLL	any block
RCM 	Remote Control Memory – RCM	62
RMC 	Remote Motor Control – RMC	136
RMSC 	Remote Manual Set Constant – RMSC	68
SOE 	Sequence of Events	99, 243
SPEC 	Specification – SPEC	any block
STN 	Control Station – STN	21, 22, 23, 80
TXT 	Text Selector – TEXT	151
TEXTSTR 	Text String - TEXTSTR	194

The common Bailey computer interfaces used to gain access to the Bailey exception report blocks presented in the above table also support the ability to allow a system to emulate and output most of these same point types. They are called “output report” block types. Bailey consoles receive the point type, exception reported data directly from the Bailey interface in the same format as if it was coming from an equivalent Bailey controller. This allows the predefined Bailey console faceplates for each of these control point types to be utilized when receiving data from the OPC90 Server report point types. Instead of addressing the tags to a block in a Bailey controller, they are addressed to a block number assigned within the Bailey interface that OPC90 is communicating. The









ring and node address to use is that assigned to the interface with a fixed module address of two.

The following table presents a list of Bailey interfaces supporting the output report block types.

<b>Bailey Interface</b>	<b>Bailey System</b>	<b>Max. Number of points</b>	<b>Supported by OPC90 Server</b>
NCIU02	Network 90	2,500	Yes
NCIU03	Network 90	5,000	Yes
NCIU04	Infi 90	10,000	Yes
INPCI02	Infi 90 / Symphony	5,000	Yes
IIMCP01	Network 90 / Infi 90 / Symphony	10,000	Yes
IIMCP02	Infi 90 / Symphony	30,000	Yes
INICI01	Infi 90 / Symphony	10,000	Yes
INICI12	Infi 90 / Symphony	10,000	Yes
INICI13	Infi 90 / Symphony	30,000	Yes
INICI03	Infi 90 / Symphony	30,000	Yes
SPIET800 (see note 1)	Symphony Plus	30,000	Yes

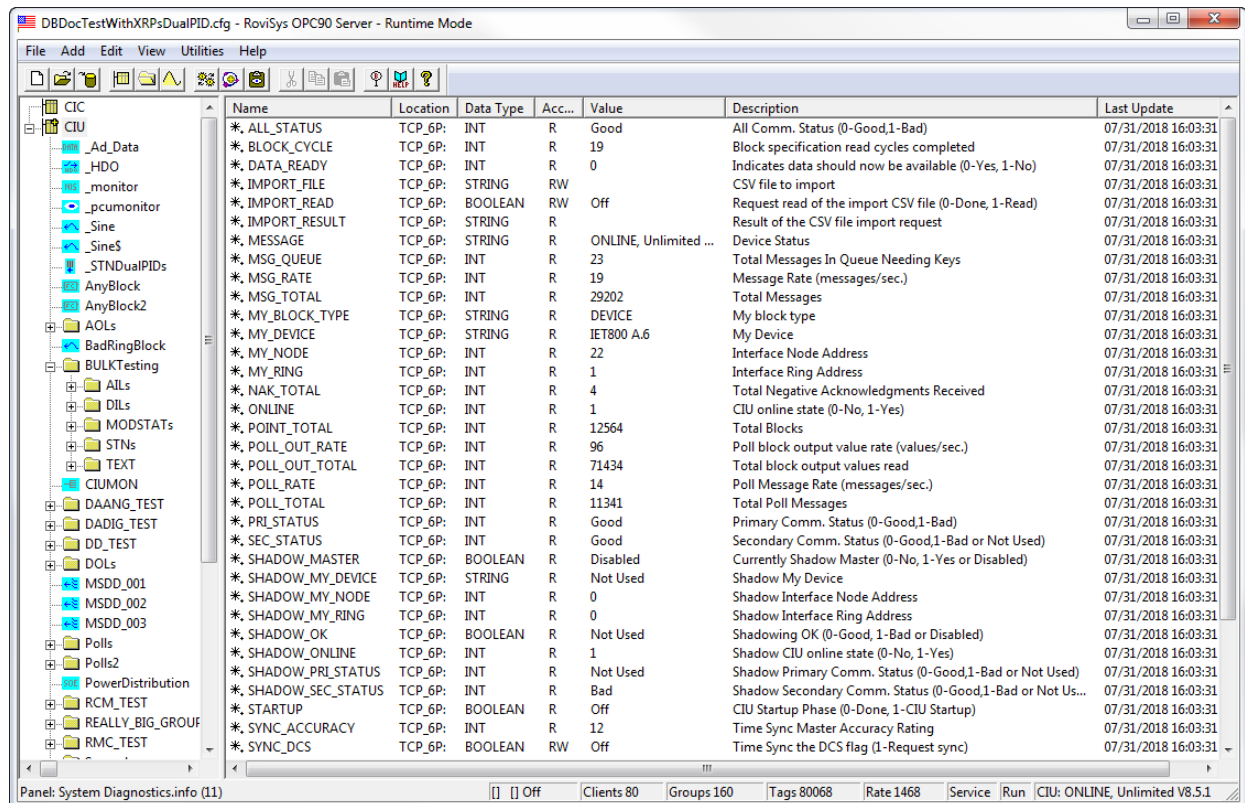
- 1.) See “Using OPC90 with IET800” for information on setting up this device.

The following table summarizes the output report block types supported by OPC90 Server. These blocks enable OPC clients to source these Bailey tag types to the Bailey system.

Data Type	OPC90 Block Type	Responses as Bailey Function Code	Description
Analog	AOL 	30	Analog Output Loop used to source an analog value to Bailey function codes 26, 121 and analog Bailey console tags.
Digital	DOL 	45	Digital Output Loop used to source a digital value to Bailey function codes 42, 122 and digital Bailey console tags.
Digital	ODD 	123	Output Device Driver used to source device driver data to a Bailey DD console tag. The Bailey console can issue on or off commands to this block with the OPC client indicating results using a set of feedback indicators.
Digital	OMSDD 	129	Output Multi-State Device Driver used to source multi-state device driver data to a Bailey MSDD console tag. The Bailey console can issue state commands to this block with the OPC client indicating results using a set of feedback indicators.
Digital	ORCM 	62	Output Remote Control Memory used to source remote control memory data to a Bailey RCM console tag. The Bailey console can issue on or off commands to this block with the OPC client indicating results using a feedback indicator.
Digital	ORMC 	136	Output Remote Motor Control used to source remote motor control data to a Bailey RMC console tag. The Bailey console can issue start or stop commands to this block with the OPC client indicating results using a set of permissive and feedback indicators.
Analog	ORMSC 	68	Output Remote Manual Set Constant used to source remote manual set data to a Bailey RMSC console tag. The Bailey console can issue set point values to this block for use by the client.
Analog	OSTN 	21, 22, 23, 80	PID control loops (OSTN stands for Output Station Control) used to source PID control data to a Bailey STN console tag. The Bailey console can issue PID control commands (set point, control output, mode) to this block with the OPC client responds back with process variable and other STN indicators.

## 5 OPC90 Server Application Menu Items

The following picture shows the OPC90 Server main menu items available to the program user for managing databases, editing the database, selecting run time activities and accessing program information along with online help.



The “File” menu item supports generation of new databases, opening an existing database, saving a database, enabling / disabling auto save, creation of database points by importing a CSV file and saving of the database in CSV file format.

The “Add” menu item supports addition of new database devices, groups and blocks.

The “Edit” menu item supports the following activities:

- deletion of database entities (devices, groups and blocks),
- definition of communication port parameters,
- definition of TCP port mapping,
- mapping of engineering unit strings,
- definition of text messages,
- setting up red tag users,
- definition of SOE point names,
- viewing of database entity properties,
- setting the program priority,
- setting up configuration shadowing,

- setting up error detection log,
- setting up how the database tree is sorted.

The “View” menu item supports the following activities:

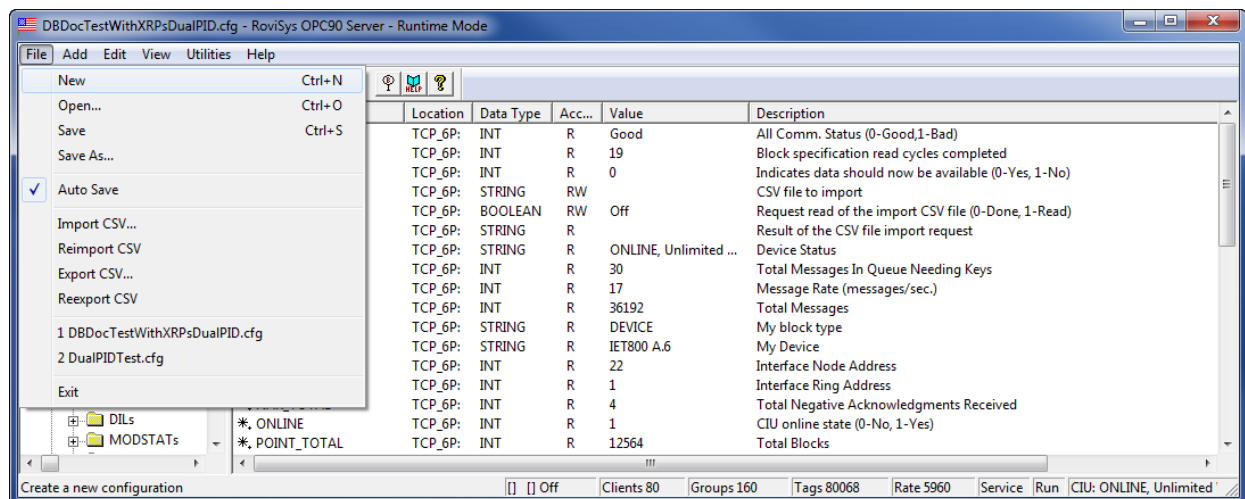
- selection of monitor mode,
- program status bar,
- starting up in run time mode,
- enabling or disabling OPC90 as a service,
- locking out OPC access,
- refreshing the database tree,
- view / set 2<sup>nd</sup> instance OPC90 activity timer.

The “Utilities” menu item can be used to run the Windows service manager or DCOMCNFG programs along with viewing OPC90 log files.

The “Help” menu item is used to find blocks, invoking online help and determine program version number.

## 5.1 File

The following picture shows the available File menu items:



### 5.1.1 New

Select “New” to create a new database. The current database will be closed. If any database entities had been changed without being saved the user is given the opportunity to save the database before completing the new database request.

### 5.1.2 Open

Select “Open” to open a database that had been previously created. The OPC90 convention is to open databases stored in the subdirectory called CFG (configuration files). A different database cannot be opened while OPC clients are attached and OPC90 is in runtime mode. Stop the OPC clients and take OPC90 out of runtime mode by

enabling and then disabling monitor mode. If OPC clients can't be stopped, use the View | Lockout OPC Access menu item to temporarily prevent their attachments.

### 5.1.3 Save

Select "Save" to save the currently opened database. The OPC90 convention is to save the database in the subdirectory called CFG (configuration files). The last saved database becomes the OPC90 working database that is automatically restored when the program is opened by the user or another OPC client. Therefore, it is important to save a newly created database to set it as the working database. OPC90 configurations can be saved using any file name and extension. The general accepted naming convention adopted by RoviSys is to use a file extension of ".CFG" to denote it's a configuration file. As just mentioned a subdirectory is generated by the OPC90 install set named CFG as the intended place to save OPC90 configurations.

### 5.1.4 Save As

Select "Save As" to save the database under a new name. This newly named database will become the working database the next time the user or OPC client starts OPC90.

### 5.1.5 Auto Save

Select "Auto Save" to enable / disable automatic database saving. Auto save **must** be enabled when using configuration shadowing. Otherwise, both OPC 90 Servers will always run as the database master. It is best to run with this feature enabled to avoid accidental loss of database changes. When enabled, the database will automatically be saved whenever OPC90 is currently communicating with the DCS. By default auto save will occur every 60 seconds whenever changes have been made to the database or new real time specification data has been received. The default auto save time can be changed when this feature is enabled. It will also keep the database current with the latest snap shot of real time values. Values such as alarm limits, ranges, spans, engineering units and others received as exception report specifications when the interface starts up and after they are tuned. These values are restored from the database when OPC90 is first started thus replacing default settings for actual system settings while waiting for interface startup to complete.

### 5.1.6 Import CSV

Select "Import CSV" to define a new database (after selecting File->New) or add additional points to an existing database. The OPC90 convention is to import CSV files stored in the subdirectory called CSV. Comma Separated Variable files (CSV) can be generated in a program such as Microsoft Excel or a text editor like Notepad. Each line in these files define OPC90 devices and blocks. See the section that deals with CSV file formats for additional information. This function will be blocked if OPC clients are currently attached to OPC90 or it is currently operating in monitor mode. If PC COM ports are being used for serial CIU communication, after importing, check the baud rate setting for those COM ports match CIU setting by using the OPC90 Edit | Com Ports.



### 5.1.7 Reimport CSV

Select “Reimport CSV” to import the last CSV file that had been previously imported. This function will be blocked if OPC clients are currently attached to OPC90 or it is currently operating in monitor mode.

### 5.1.8 Export CSV

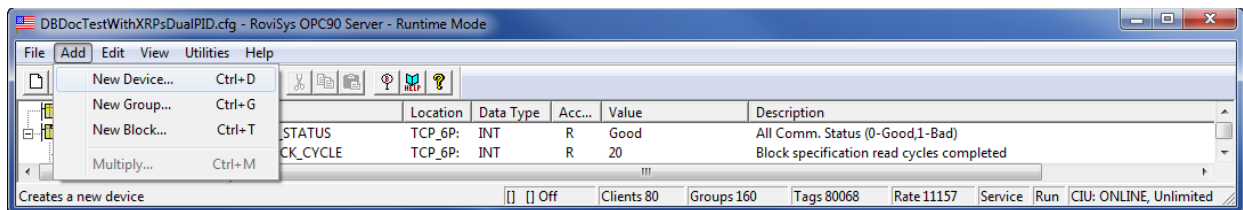
Select “Export CSV” to save the current database in Comma Separate Variable format. The OPC90 convention is to store these files in the CSV directory. This function is useful for saving the database in a format that can be later imported after making “bulk” editing changes to it or when upgrading from one version of OPC90 to another where additional tags have been defined for any given OPC90 block type. See the section entitled “Trouble Shooting Hints - Cannot See all documented block attributes” for further discussion.

### 5.1.9 Reexport CSV

Select “Reexport CSV” to save the current database in CSV format under the previously exported name. This feature is like a fast save.

## 5.2 Add

The following picture shows the available Add menu items:



Select “New Device” to add a new device to the database. A database must have at least one device definition. The name given to the device becomes the first part of the OPC tag name used by OPC clients when accessing a piece of OPC data from OPC90. A device defines operational characteristics that OPC90 uses when communicating with the physical Bailey interface, often generically called a CIU (Computer Interface Unit). One of the most important communication characteristics that must be considered is the physical PC communication channel that has been attached to the CIU. Typically this channel is an RS232 port that must be setup using the Edit->Ports menu item. For the INICI03 interface the communication channel could be RS232 or SCSI. When it is a SCSI channel, no additional communication setup is required beyond selecting the SCSI port that has been cabled to the INICI03 INICT03A module. See the DEVICE block for more details.

Select “New Group” to defined a folder under a device in which collections of other OPC90 groups and blocks can be configured. Groups allow data organization within the OPC90 database. They are not required but if used the name given to the group

becomes part of the OPC tag name used by OPC clients when accessing OPC data from OPC90.

Select “New Block” to add an OPC90 block to a device or in a device group. OPC90 blocks are given a name that typically identifies its purpose and becomes part of the OPC tag name used by OPC clients when accessing OPC data from OPC90. The block type defines what type of data to expect from the Bailey system for the given block and the address at which it resides.

The “Multiply” selection allows bulk duplication of the last selected block to be added to the database. It has limited use in real world applications, most typically used to automatically generate a series of output type blocks suitable for OPC client programmatic generated output values.

### 5.2.1 Device

When the add device item is selected the following dialog is displayed:

**Device Properties**

Name:  Description:

**COMMUNICATION PARAMETERS**

Scheme:

Primary Port:

Secondary Port:

**TIME SYNCHRONIZATION**

☒ Infi 90 ☐ Network 90

☒ PC Set DCS time ☒ Use DCS time stamp

☒ PC Get DCS time ☒ Allow DCS time stamp override

Accuracy rating ☐ Y2k time warping

**DEVICE PARAMETERS FOR IET800 A.6**

Max out block # (AOL, DOL, DSTN, etc)

Watchdog timer (in 2.5 second counts)  msecs

Import exception report request rate  msecs

Export exception report send rate  msecs

Block specification read period (seconds)

Max block specification reads per period

Max keys per PCU (0 = device defined)

Max GMI per second per PCU node

☒ Enable turbo polling ☐ Non-keyed messaging

☒ Screen exception reports ☐ Enhanced analog precision

☐ Establish points online ☒ MSDD pulse out handling

**SUPERVISORY CONTROL WRITE LOCKS**

☐ Lock out all operator write requests (from OPC clients)

**SIMULATION**

☐ Enable ☒ Device and input blocks

☐ AOL and DOL block inputs  % factor

**TAG CONDITIONS**

Startup

☐ Good ☐ Startup complete after data received

☐ Uncertain ☐ Auto disconnect unused blocks

☒ Bad ☐ Restore last analog and digital tag values

☒ Set bad quality on max exception timeout

☒ Any input write applies to all output blocks

☐ Allow turbo reads for bad quality blocks

**STN BLOCKS**

Fast update for  minutes every  msecs

☒ Set .KFAST on .K\* tag writes

☐ Automatically send CPU\_OK

☐ Represent mode as 0 - 5

**MISCELLANEOUS**

☐ Continuous OPC client updates ☒ Group output writes

☐ XRP write confirmation

Days to retain debug logs:

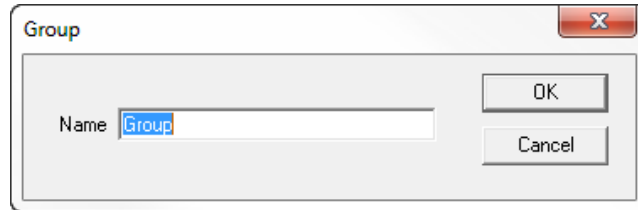
Days to retain SOE logs:

A device name should be defined that usefully identifies it. The device name becomes part of the OPC tag name. An optional device description can be defined to provide additional device information. For most installations the default settings for the device

properties are appropriate. See the DEVICE block description for further details on each of these settings.

### 5.2.2 Group

When the add group item is selected the following dialog is displayed:



A group name can be defined describing the purpose of the blocks configured within the group. The group name becomes part of the OPC tag name.

### 5.2.3 Block

When the add block item is selected the following dialog is displayed:

**Create a New OPC90 Block**

Name:  OK Cancel

Description:

Block Type:  ☒ Validate Type ☐ Read only

**ADDRESSING**

Ring:  Node:  Module:  Block:

**Logic State Descriptors**

	Zero State	One State
Output	<input type="text" value="Off"/>	<input type="text" value="On"/>
Feedback 1	<input type="text" value="Off"/>	<input type="text" value="On"/>
Feedback 2	<input type="text" value="Off"/>	<input type="text" value="On"/>
Feedback 3	<input type="text" value="Off"/>	<input type="text" value="On"/>
Feedback 4	<input type="text" value="Off"/>	<input type="text" value="On"/>
Permissive 1	<input type="text" value="Off"/>	<input type="text" value="On"/>
Permissive 2	<input type="text" value="Off"/>	<input type="text" value="On"/>

**Alarm Hold**

milliseconds

% of span

State 0:  State 1:   
 State 2:  State 3:

**Analog Output Parameters**

Sig. Change:  %

EU Code:

Init Value:

Zero:

Span:

Lo Alarm:

Hi Alarm:

Dev Lim:

STN Type:

**Poll Block Parameter**

Poll Interval:  milliseconds

**Digital Output Parameters**

Initial Value:

Alarm State:

Faceplate Type Code:

**Exception Report Outputs**

Max Time:  seconds

This dialog is used to add all block types supported by OPC90. A block name must be defined which becomes part of the OPC tag name used to access the various block data tags. An optional block descriptor can be entered that documents the purpose of the block. The block type list box allows selection of the specific block type being defined. The addressing section is used to define the address within the Bailey system that provides the block data. The other field definitions will be enabled based on the block type selected. For example when an AIL block is defined the "Alarm Hold" section is enabled to define the hold criteria. See the blocks section for further details on each specific setting.

When adding a block while OPC90 is currently communicating with the ABB Bailey system, the "validate type" check box will also be displayed. It is best to leave this validation enabled since it will verify the block type being added actually exists within the ABB Bailey system.

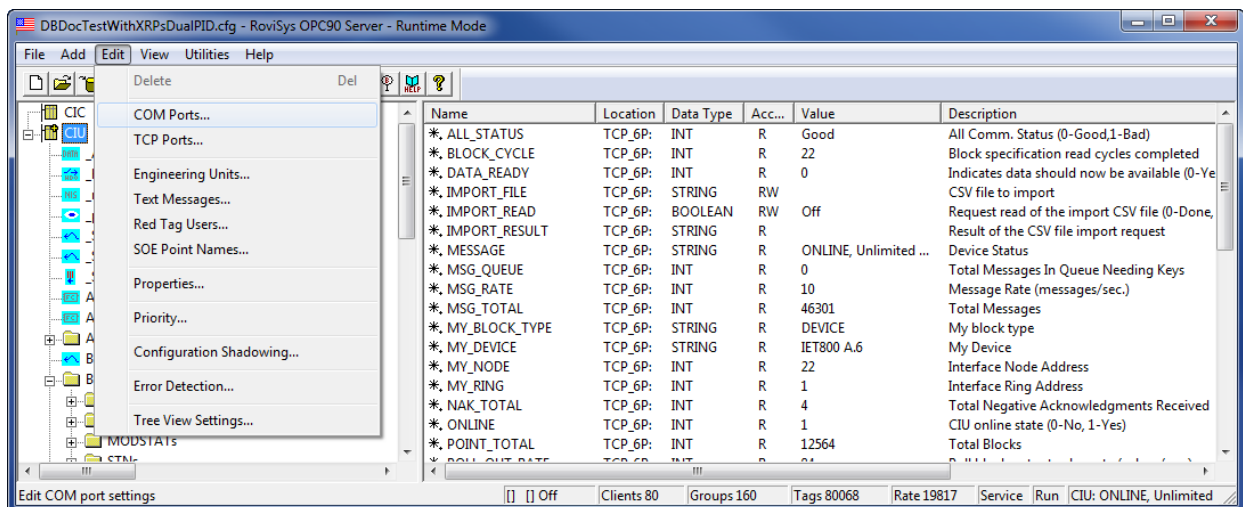
Block types that receive data from the ABB Bailey system will also show the “read only” check box. When set it prevents any type of write to the block that causes a change in the ABB Bailey controller. It allows individual blocks to be marked as read only instead of using the global lock found in the Device block properties.

### 5.2.3.1 Alarm Hold

A recent new feature is the ability to define an “Alarm Hold” for AIL and DIL blocks. The alarm hold causes the AIL and DIL blocks to keep the alarm active tag set after the alarm condition has ended. For a DIL block the amount of time it stays set is defined by the hold time. For an AIL block it’s the hold time OR the value returning out of the alarm level by the specified deadband. The deadband is defined as percent of the AIL block span received from the ABB Bailey system. Note that the default settings for alarm hold time and deadband are zero, which also disables the feature. Alarm hold is useful for filtering nuisance alarm transitions.

## 5.3 Edit

The following picture shows the available Edit menu items:



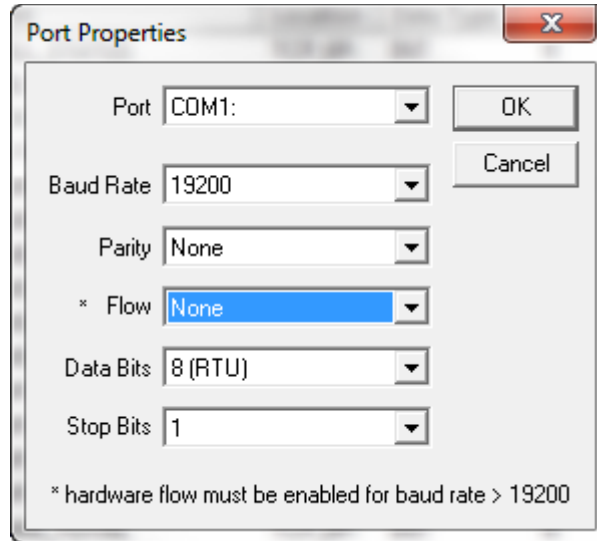
### 5.3.1 Deletion

Select “Edit | Delete” to delete the selected block, group or device. Be careful when selecting this item. If a group is deleted, all blocks under that group are included in the deletion. If a device is deleted, all groups and blocks under that device are included in the deletion.

Device deletion not allowed when in runtime mode of operation. Block deletions are allowed in runtime mode of operation even when the block is providing data to connected OPC clients. For this case, the block is deleted and the data being sent to OPC client set to bad quality, not connected. This rule is enforced across shadowed databases. A block deleted on a shadowed setup also gets deleted on other shadowed PC.

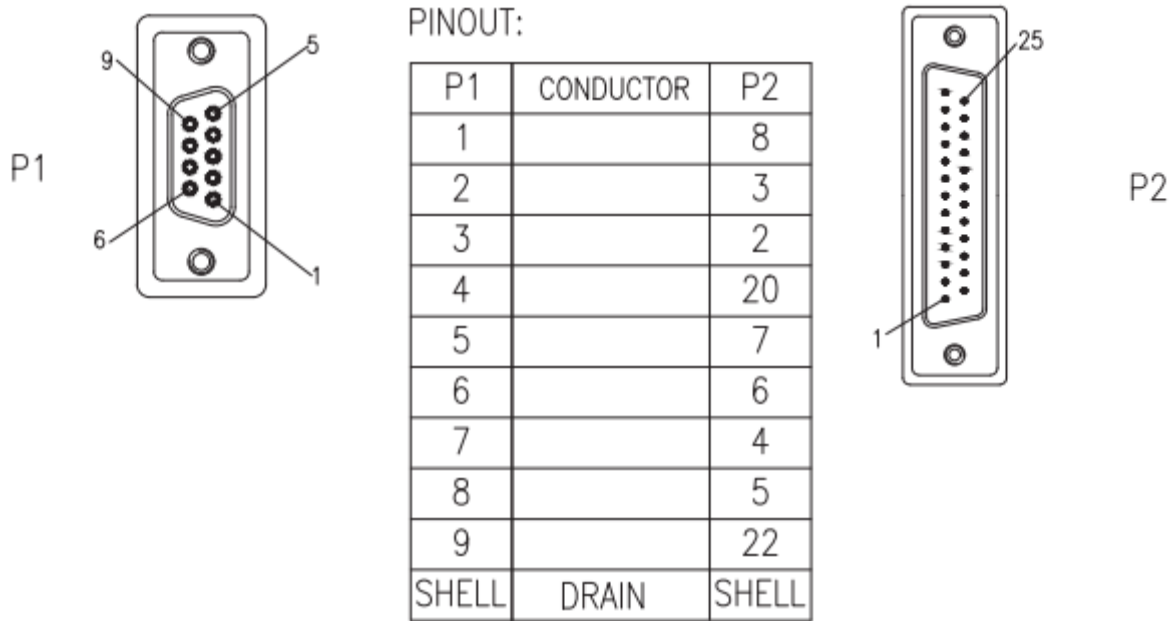
### 5.3.2 Com Ports

Select “Edit | Com Ports” to define the communication characteristics of any given PC COM port assigned to a device. When this item is selected the following dialog is displayed:



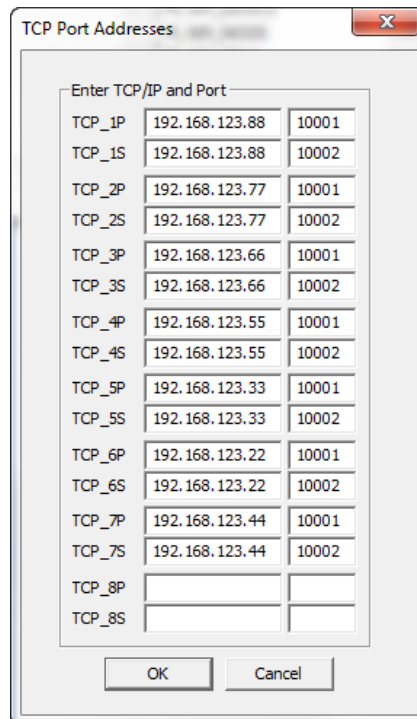
OPC90 uses the communication characteristics setup with this item and not those configured under the windows global port settings. Most Bailey interfaces utilize the settings shown by this dialog. The exception is the CIC module which has a maximum baud rate capability of 9600 baud.

Serial cables typically used between the COM port and ABB Bailey termination unit are DB25 to DB9 serial modem cables – M/F. Startech.com part number MC6MF is a good example of this cable available from CDW or other online resellers.



### 5.3.3 TCP Ports

Select “Edit | TCP Ports” to edit the 16 available OPC90 TCP ports. When this item is selected the following dialog is displayed:



TCP Port Addresses

Enter TCP/IP and Port

TCP_1P	192.168.123.88	10001
TCP_1S	192.168.123.88	10002
TCP_2P	192.168.123.77	10001
TCP_2S	192.168.123.77	10002
TCP_3P	192.168.123.66	10001
TCP_3S	192.168.123.66	10002
TCP_4P	192.168.123.55	10001
TCP_4S	192.168.123.55	10002
TCP_5P	192.168.123.33	10001
TCP_5S	192.168.123.33	10002
TCP_6P	192.168.123.22	10001
TCP_6S	192.168.123.22	10002
TCP_7P	192.168.123.44	10001
TCP_7S	192.168.123.44	10002
TCP_8P		
TCP_8S		

OK Cancel

As can be seen by this dialog, OPC90 supports definition of up to 16 predefined TCP names. Definition of each name needs the TCP address of the Ethernet to Serial device

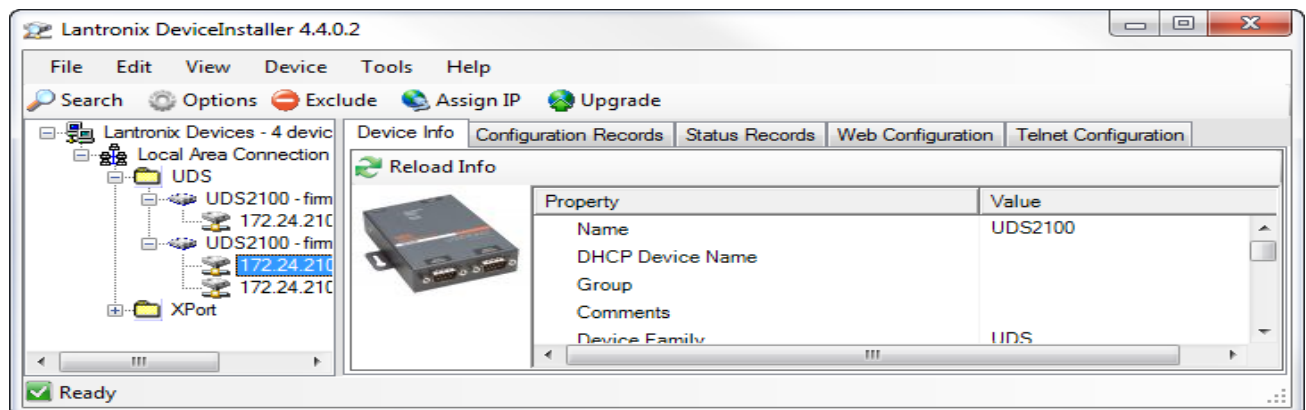
and port number to attach with on that device. The intention of the TCP\_xP and TCP\_xS names indicate primary and secondary ports. These are useful when using the DEVICE block redundant interface and dual interface communication scheme. Any TCP name can be used with any DEVICE block port. The important thing is to pick one that corresponds with the correct TCP address. The naming convention of using the 'P' and 'S' is only for organization purpose.

It is **important** to realize that OPC90 will setup a socket connection to the Ethernet to Serial device using the addressing information associated with the chosen TCP name. The software provided with the Ethernet to Serial device sets up the addressing information. Specifically setup the following items.

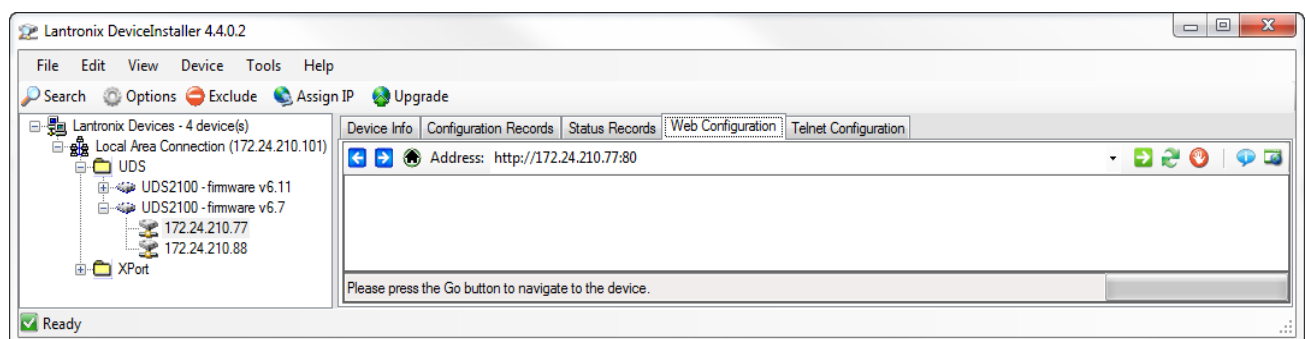
- Fixed IP address,
- Default baud rate of each serial port (must match CIU setting),
- Serial port set for 8 data bits, 1 stop bit, no parity and no flow control.

#### 5.3.3.1 Setting Up Lantronix UDS 2100

Setup of the UDS 2100 is accomplished using the Lantronix DeviceInstaller software. Run DeviceInstaller and click on the UDS 2100 device to be setup. It looks as follows.

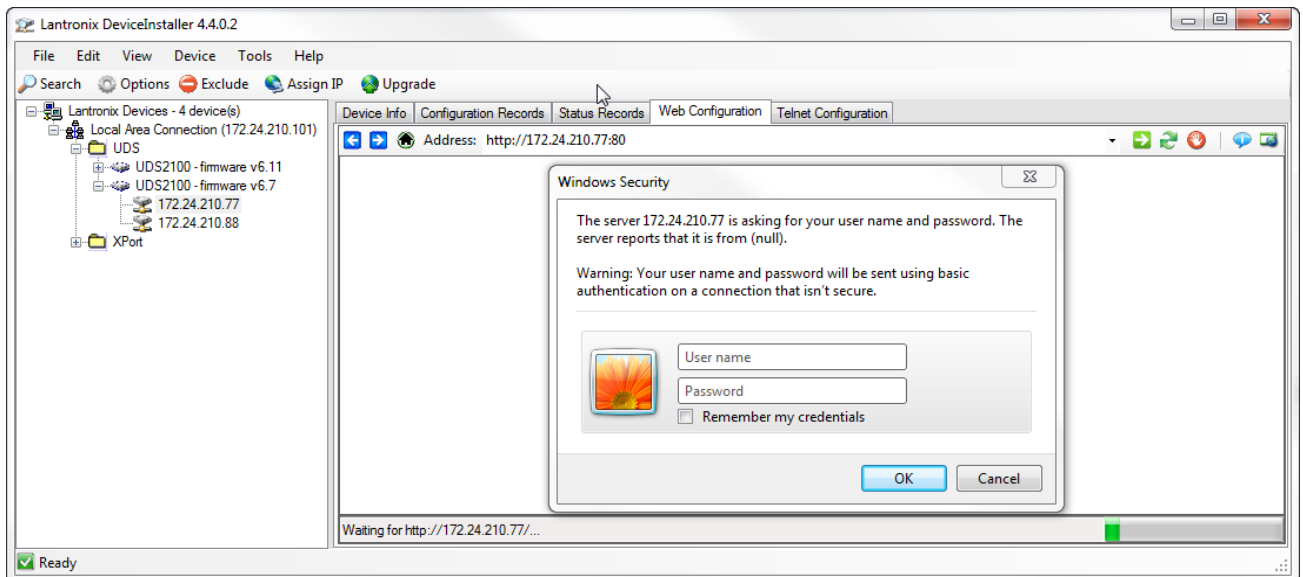


Its web interface tab is used for the setup. Click on the "Web Configuration" tab. It looks as follows.

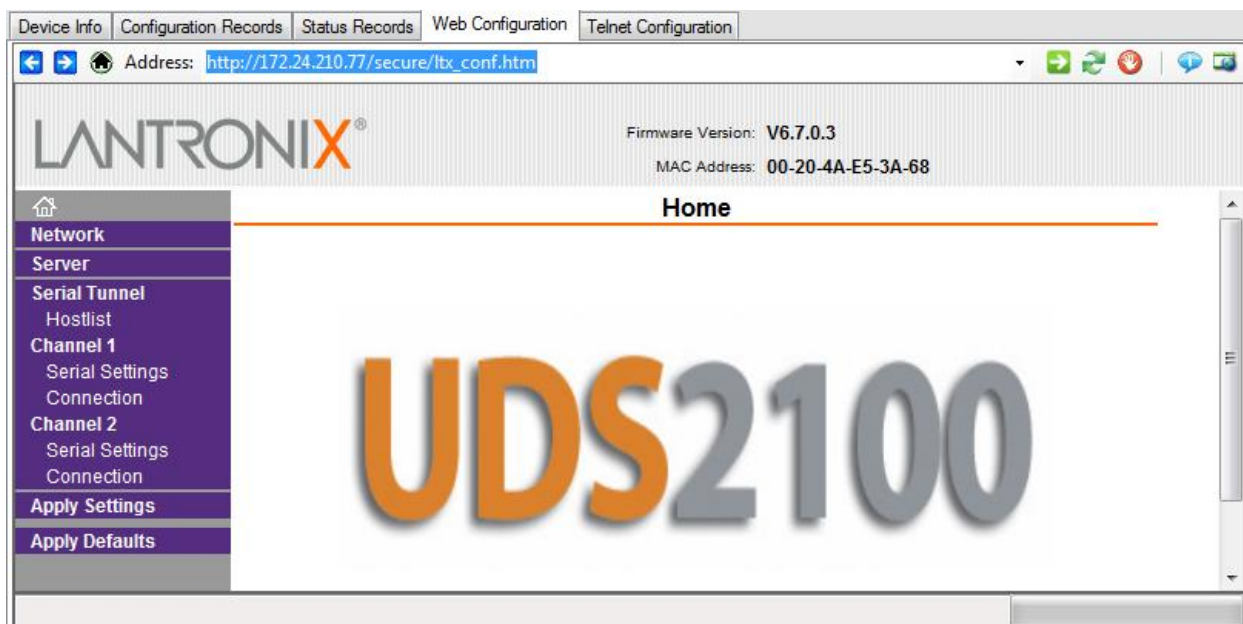




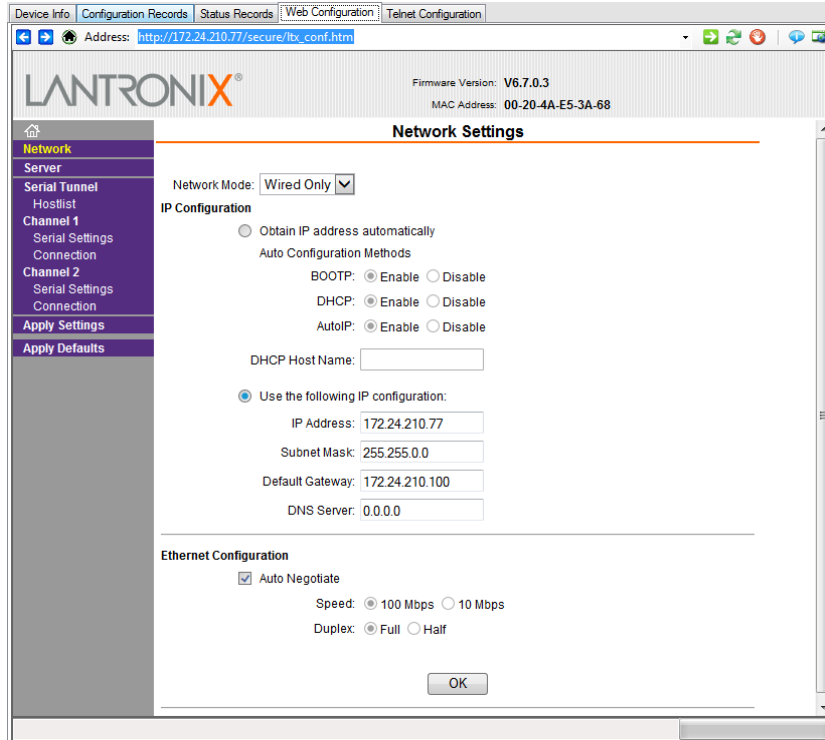
Click on the white arrow in the green box that is pointing to the right (in this example located right of the `http://172.24.210.77:80` URL). It looks as follows.



No user name or password is required so just click the OK button in the login dialog and it will display the following setup information.

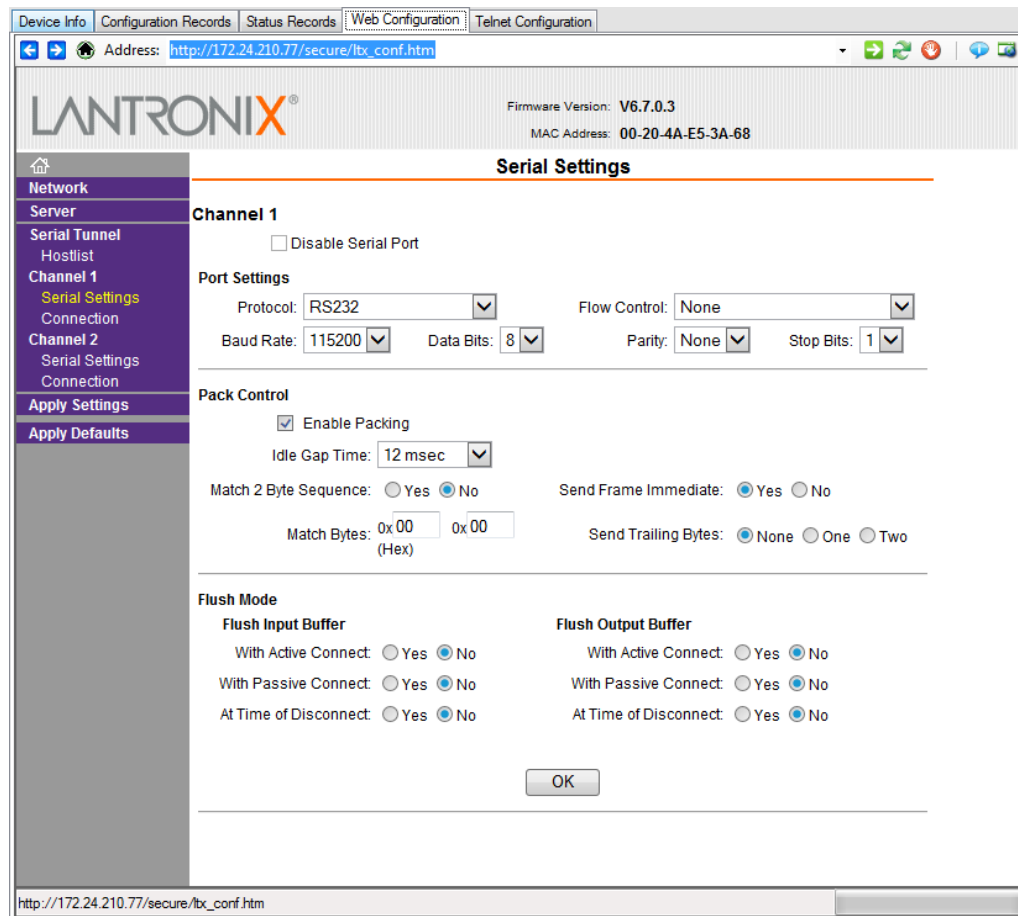


Click on “Network” to setup a fixed IP address for the UDS 2100. It will display the following information. Pick an IP address that is consistent with the network. Obtaining an IP address via DHCP is not recommended. It could change over time which would prevent OPC90 from accessing the device. Enter the data and click the OK button.



Now it's time to setup the serial configuration data. Click on "Serial Settings selection underneath Channel 1. Setup the port settings shown as follows. Note the "Baud Rate" must be set to match the baud rate of the CIU device. In this example its set for 115,200 which is the maximum setting supported by an IET800 CIU type. All other CIU types have a maximum supported setting of 19,200. For best throughput it is recommended to use whatever maximum baud rate setting the CIU device supports.

It's **important** to enable the "Enable Packing" and "Send Frame Immediate" parameters. This significantly improves throughput for high speed serial devices such as the IET800.



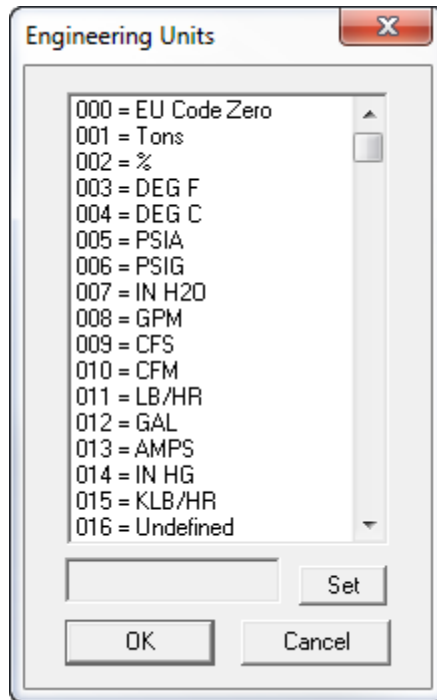
Click OK and then repeat this setup for “Serial Settings” under Channel 2.

Now it is the **important** moment to commit the configuration data to the UDS 2100. This is accomplished by clicking “Apply Settings” (just to the left where it says Pack Control). This causes the UDS 2100 to save the changes and reset itself to put them into operation.

Mission accomplished!

### 5.3.4 Engineering Units

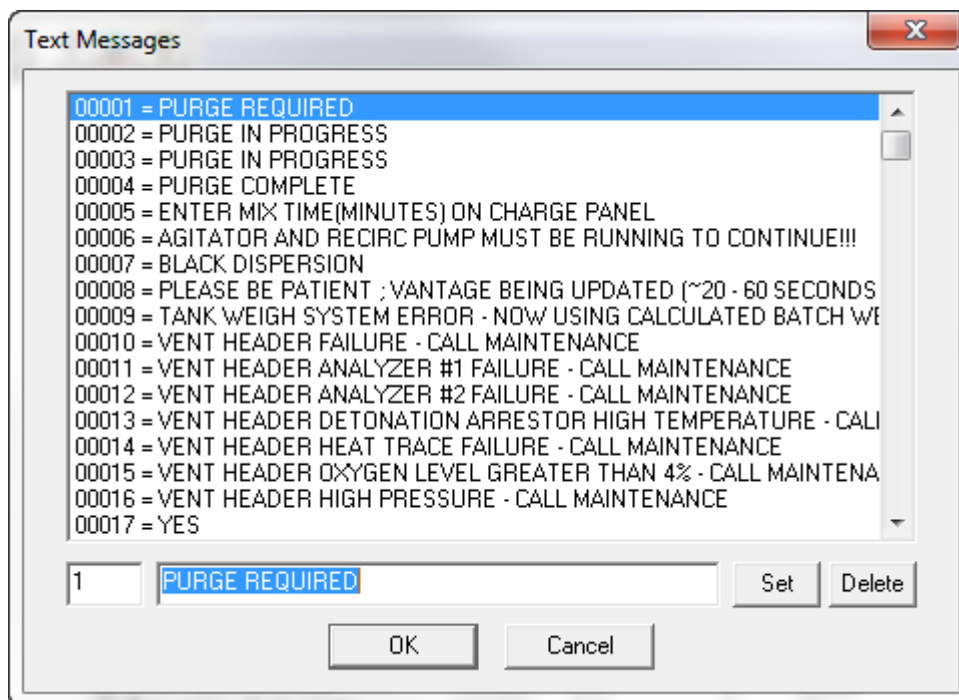
Select “Edit | Engineering Units” to defined Bailey engineering unit code to engineering string mapping. The various analog values received from the Bailey system are already in a particular engineering unit. Those blocks responsible for returning exception reported analog values also identify the engineering units using a predefined Bailey code in the range of 0 – 255. The OPC90 blocks designed to return these values also include two OPC tags called EU and EU\_TEXT. The EU tag returns the Bailey EU code associated with the analog value. The EU\_TEXT tag returns the Bailey engineering units code converted to a string value. Bailey predefines the first 16 codes. The following dialog is displayed when this menu item is selected:



Use this dialog to define the mapping between Bailey engineering unit codes and the string to be associated with each code. (Note that engineering unit mapping can also be defined using the “Import CSV” program feature.) The map is global to all OPC90 blocks that return exception reported analog values. Use of the EU\_TEXT OPC tag is optional. Its purpose is to simplify OPC client display generation and allow the source of engineering unit definition to remain singularly at the Bailey module level. Note that engineering unit code to string mapping is changeable online while OPC90 is providing data to attached OPC clients. The maximum string size for an engineering unit code is 16 characters.

### 5.3.5 Text Messages

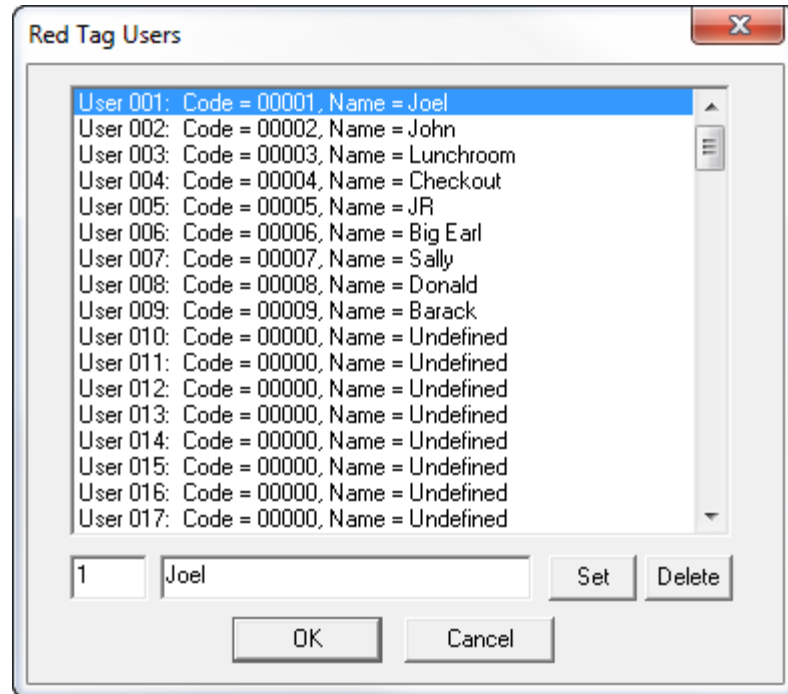
Select “Edit | Text Messages” to defined Bailey text message number to text message string mapping. (Note that text message number mapping can also be defined using the “Import CSV” program feature.) The Bailey TEXT block (F.C. 151) exception reports text message numbers that can be received and reported as OPC tags with the OPC90 TEXT block. This block provides both a message number tag called MSG\_OUT and message string tag called MSG\_OUT\_TEXT. When this menu item is selected the following dialog is displayed:



The text message number to string value mapping is global to all OPC90 TEXT blocks. Use of the MSG\_OUT\_TEXT OPC tag is optional. Its purpose is to simplify OPC client display of Bailey text messages. Note that text message mapping is changeable online while OPC90 is providing data to attached OPC clients. The maximum string size for a text message is 132 characters. Up to 10,000 message numbers can be mapped.

### 5.3.6 Red Tag Users

Select “Edit | Red Tag Users” to define users for the Bailey red tag system. OPC90 supports the definition of 128 red tag users. This dialog is used to assign a user code to a user name. (Note that red tag user mapping can also be defined using the “Import CSV” program feature.) The Bailey red tag system allows users to mark blocks to be in the red tag state. The Bailey red tag system supports up to 3 simultaneous users to have a given block marked in the red tag state. Within the Bailey system, red tag users are managed as user codes in the range of 1 to 65535. This dialog allows these codes to be assigned user names. Use of this name mapping is optional. When a red tag code is encountered that is not mapped to a name, the code will be reported instead of a name. When this menu item is selected the following dialog is displayed:



Use the list box to select a specific user (1 – 128) to be defined. The user code and name can be edited in the fields below the list box. Use the tab key to signify entry of a new value. Click on the set button to set the user code / name. Click on the delete button to delete the user code / name definition. Red tag user definition can also be accomplished using the OPC90 import feature. Consult the “CSV file format” section for specific information on how to define red tag users.

OPC90 includes three OPC tags associated with managing red tags. These tags are “.RED\_TAG”, “.RED\_CMD” and “.RED\_USERS”. These tags are valid for the DAANG, DD, MSDD, RCM, RMC and STN blocks.

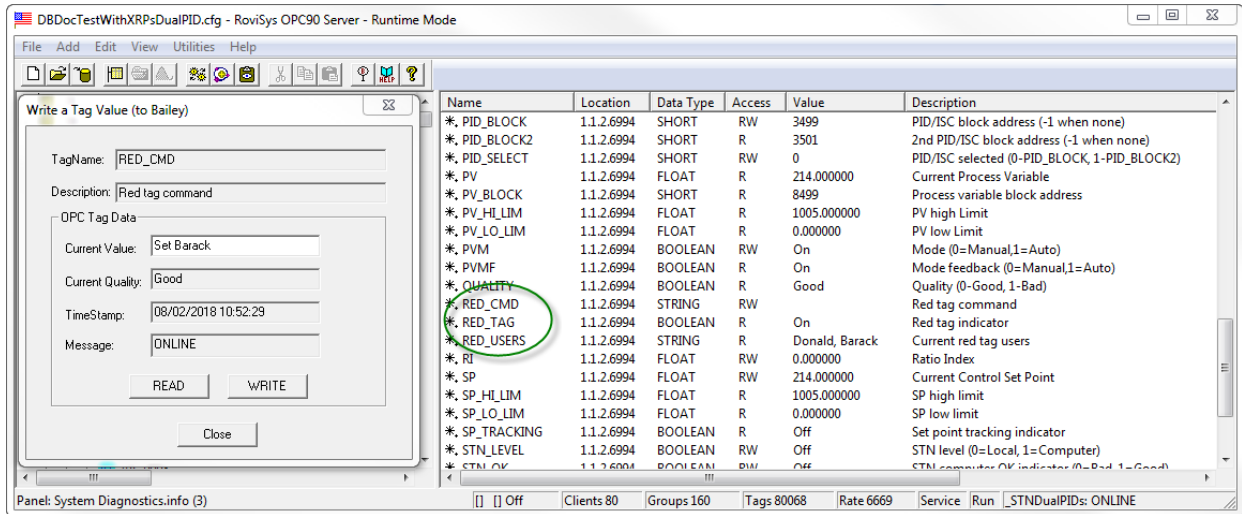
The “.RED\_TAG” OPC tag reports whether or not the block is currently in the red tag state. When set, the block is red tagged. This tag value is an exception reported value.

The “.RED\_USERS” OPC tag reports the users that currently have the block in the red tag state. As previously mentioned, up to three users can have the block red tagged. This is a string tag and will report the users separated by commas. If a user code is returned that has not been mapped by this dialog, the user code will be reported instead of a name. The red tag user codes are not exception reported and must be polled. Whenever a block is in the red tag state, the user codes will be polled once per minute. This allows OPC90 to determine red tag users generated outside of OPC90 such as from a ABB Bailey operator console.

The “.RED\_CMD” OPC tag is used to command red tag users to attach and detach with the block. The format of the command is simple. The command name is followed by the red tag user name or code. The commands for attaching red tag users are ON, SET or TAG. The commands for detaching red tag users are OFF, RESET or UNTAG. A space

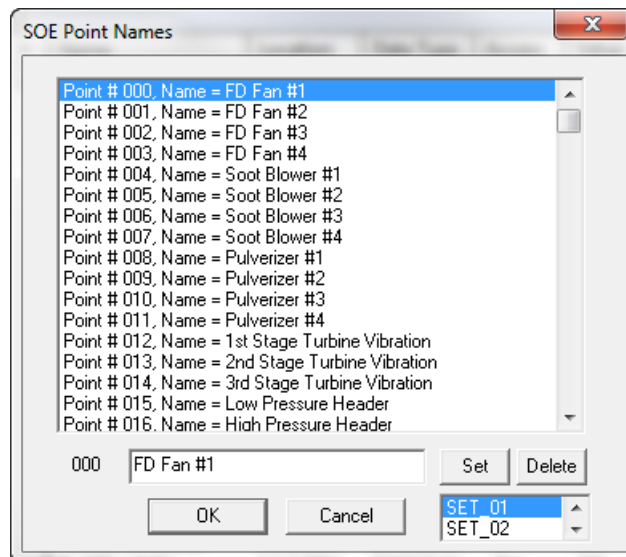
must be included between the command and user name or code. Command entry including the user name is not case sensitive. The result of the command is returned in this tag.

The following example shows the three tags associated with red tagging along with the results of the set command.



### 5.3.7 SOE Point Names

Select "Edit | SOE Point Names" to define sequence of event point names referenced by the OPC90 SOE blocks. Definition of four sets of SOE point names is supported. Each set defines 512 points. When SOE data is received from the ABB Bailey system, that data identifies points by their number (0 – 511). Those numbers are associated with point names defined by this dialog. The various SOE files generated by OPC90 contain both the point number and point name. When this menu item is selected the following dialog is displayed:





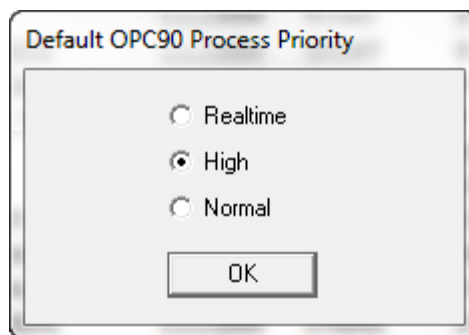
Select the current set (SET\_01 to SET\_04) to be defined using the list box at the bottom right of the dialog. The current point definition for that set is displayed in the large list box. The point number and name is displayed. Click on a point number to be edited. Its point number and name are displayed at the bottom of the dialog. Type in the point name and click the “Set” button to define that point. Use the tab key to signify entry of a new value. Click the “Delete” button to undefined the selected point. Point names must be unique within each set. SOE point definition can also be accomplished using the OPC90 import feature. Consult the “CSV file format” section for specific information on how to define SOE point names.

### 5.3.8 Properties

Select “Edit | Properties” to edit the current definitions setup for the selected device, group or block. The appropriate properties dialog will be displayed based on whether the device, group or block is the last selected database entity.

### 5.3.9 Priority

Select “Edit | Priority” to edit the default process priority that OPC90 will run. When this menu item is selected the following dialog is displayed:



As can be seen by this dialog, available priority selections are “Normal”, “High” and “Realtime”. OPC90 switches itself to the new priority as soon as the selection is made. It is not necessary to exit OPC90 and run it again for a new priority to be established. Note that when running OPC90 as a service and a second instance is attached to that service, only the priority of the service is changed. The second program instance always remains at the “Normal” priority. Installations having larger databases or OPC clients running at a higher priority than “Normal” might realize an increase in performance by raising the priority of OPC90.

### 5.3.10 Configuration Shadowing Between Redundant Servers

The configuration shadowing feature can be used to automatically keep the database and other program features synced between two different PCs running OPC90 as redundant servers. It also manages CIU online states especially important when same addressed CIUs are setup for redundant OPC90 servers (see more on this feature at the end of this section).



*It is important to setup the two redundant OPC90 servers with the same database file before turning configuration shadowing on. This is easily accomplished by copying the database file from one OPC90 PC to the other and opening that file up in the other PC to set it as the working database. The database files are found in the C:\Program Files (x86)\OPC90 Server\CFG directory.*

*Configuration shadowing must only be used between OPC90 servers of the same revision level. Unpredictable results will occur when shadowing between two unlike software revision levels.*

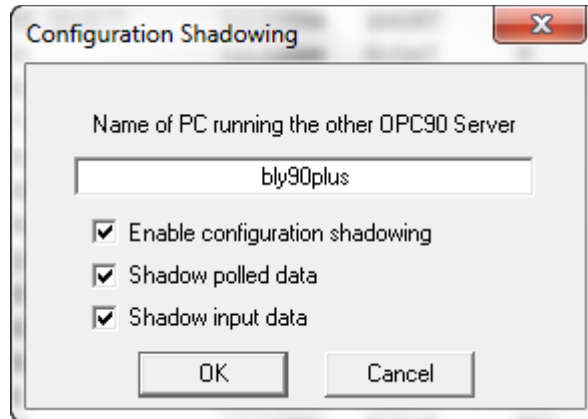
*Configuration shadowing may not work when the redundant OPC90 servers are running as a service from the system account. If the connection status indicates [ ]< >[ ] Bad (shown in the program status bar) the service must be changed to run from a privileged user account. Whether to use system account or privilege account is dependent on how operating system network service security is setup so trying it either way may be required.*

*IMPORTANT! If configuration shadowing remains bad on either PC make sure file sharing is enabled on both PCs. Also check the Windows local security policy for the existence of a “Network access: Restrict anonymous access to Named Pipes and Shares” policy. If it exists, make sure it is disabled on both PCs.*

*When operating in runtime mode, it is important to make database changes using the current OPC90 running as shadow master. This can be determined by looking at the device block “SHADOW\_MASTER” tag while in monitor mode, device block faceplate or OPC90 program window title.*

Configuration shadowing works with the File Import and Export features, and all selections available from the Add and Edit menu selections. Its purpose is to minimize the engineering effort required to maintain redundant OPC90 servers. The File | Auto Save feature is not shadowed and must be individually setup for each server. Auto Save MUST be enabled on both shadow servers for configuration shadowing to work properly.

Select “Edit | Configuration Shadowing” to enable / disable configuration shadowing. When this menu item is selected the following dialog is displayed:



Type the name of the other PC that is running OPC90. Configuration shadowing is enabled when the “Enable configuration shadowing” checkbox is checked. It is disabled when unchecked.

The “Shadow polled data” option should normally be checked. This enables the PC running OPC90 as the shadow secondary to acquire all polled data from the PC running OPC90 as the shadow master. This feature keeps polling load on the ABB Bailey PCU nodes limited to a single instance of redundant OPC90 servers instead of both polling for the same data.

The “Shadow input data” option should be checked when output type blocks are configured (AOL, DOL, ODD, OMSDD, ORCM, ORMC, ORMSC and OSTN) and the OPC client that writes to these blocks does not implement hot redundancy. This means it only writes input values to the output type blocks on the master OPC90 and not the secondary. With this option set, the input values being written to the master OPC90 will also be sent by the master to the secondary OPC90. This allows the secondary OPC90 to take over using the latest OPC client written values.

The state of configuration shadowing is displayed in the OPC90 program window status bar as follows:

☐ ☐ Off  
☐ ↔ ☐ Good  
☐ < ☐ > Bad  
 Bad Versions

The Device block also has tags that can be accessed by OPC clients to monitor the state of configuration shadowing. See the Device block for more details of what can be monitored.

Configuration shadowing has been designed to keep the standard OPC90 ABB, CFG, CSV, SOE and PANELS directories synced between two OPC90 servers running on different PCs. These are sub-directories of the C:\Program Files (x86)\OPC90 Server\

directory (or the alternative path setup during program installation). Therefore, if the user stores files outside of these locations, configuration shadowing should not be used.

In addition to keeping the above mentioned sub-directories synced with each other, configuration shadowing also copies runtime changes to the database such as block modifications, additions and deletions. So for example, when blocks are added using the file import feature, those blocks are also added to the other OPC90 server database.

Configuration shadowing will dynamically determine which OPC90 is operating as the “Shadow Master” and “Shadow Secondary”. Data such as the CIU communication status, state (online or standby), number of current OPC connections and OPC90 start time determine this role.

Determination of the role goes like this. The OPC90 with the most communicating CIUs is the master. If they are equal, the OPC90 with the most CIUs online is the master. If they are equal, the OPC90 with the most OPC connections is the master. If they are equal, the OPC90 that started up first is the master. These factors are being determined as part of startup and shared between the two OPC90s. This means some of these factors change as part of startup and therefore the shadow roles may swap back and forth between the two OPC90s a couple of times when starting up. This is normal and expected, especially when using same address CIU setups. Quickly the roles will lock in and stay that way until something changes with the communication state between OPC90 and the CIUs.

The Device block has a tag that reports the current role. Although no adverse effects will occur if editing is done on the shadow secondary, it is best to do this on the OPC90 currently operating as the shadow master. The current shadow state can also be determined from the Device block faceplate and OPC90 program window title.

#### 5.3.10.1 *Redundant Same Address CIUs*

Configuration shadowing shares the communication state and address of CIUs between the two OPC90 servers. This data is useful when using OPC90 AOL, DOL, ODD, OMSDD, ORCM, ORMC, ORMSC and OSTN blocks. Nodes within the ABB Bailey system that need to receive data from these blocks do not need to be concerned with getting the block data from two different CIU addresses. The redundant OPC90 servers can be setup with CIUs having the same address. Only one CIU will be commanded online. The other will have the database downloaded to its CIU and ready to be commanded online when communication with the other OPC90 server fails or the other OPC90 server communication with its CIU fails. *For this type of setup, it is **important** to configure the OPC90 Device block watchdog time to a non-zero value. Settings in the range of 2-4 are common.* See the device block for more information about the watchdog time.

Note that a network failure will cause the backup OPC90 server to command its CIU online if configuration shadowing cannot communicate with the other OPC90 server. By design, this will occur between one and two minutes after the network failure occurs. If

the other CIU is still online, the backup being commanded online will fault with a code indicating a same address was detected. Therefore, don't conclude that CIU red light conditions are always the result of faulty hardware. Make sure network interruptions are not occurring.

If configuration shadowing is turned off for systems setup with same address CIUs while the standby OPC90 is communicating with the CIU, it will command its CIU online resulting in the aforementioned CIU red light condition to occur. The only way to avoid this from happening is to disconnect all OPC clients from OPC90 and command it out of runtime mode of operation before turning off configuration shadowing. If the red light condition does occur, after configuration shadowing is re-enabled, just physically reset the red lit CIU to put it back into operation.

#### 5.3.10.2 *OPC Client Access of Redundant Servers*

With redundant OPC90 Servers, the OPC client must implement logic to decide which server to access the data from. A common method is to base the decision on two OPC90 data points.

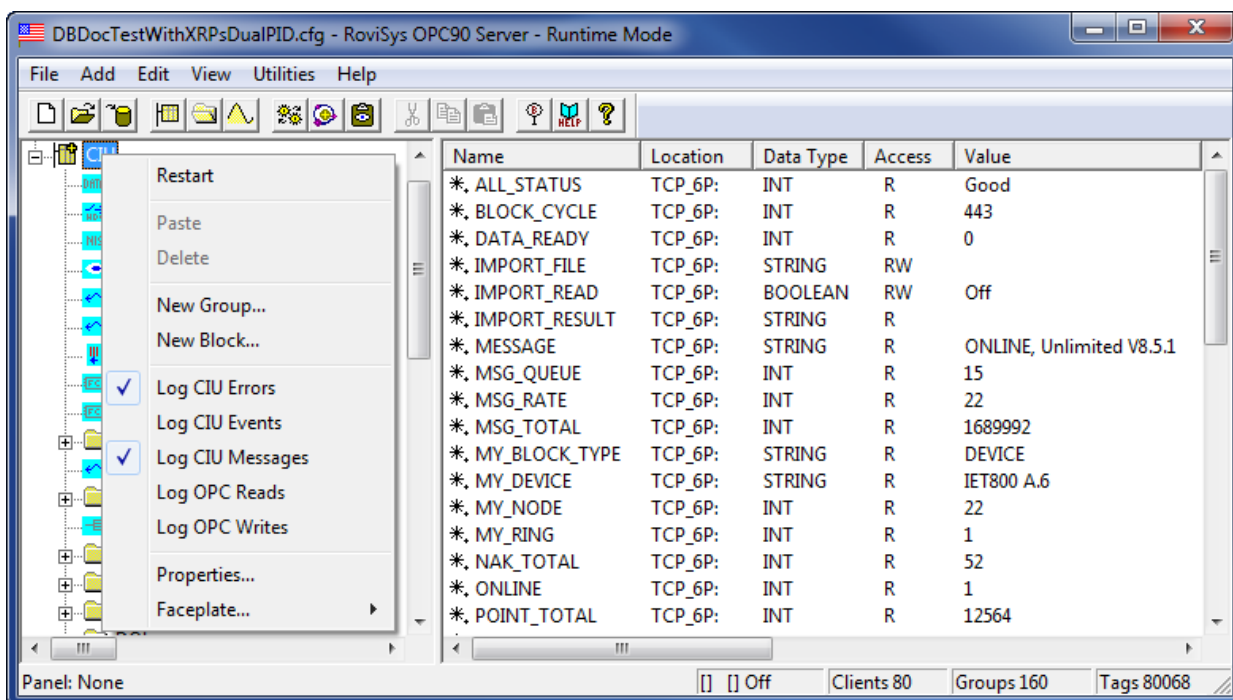
The first is the state of the OPC90 Device block ALL\_STATUS tag. When the value of this tag is zero, OPC90 communication with the CIU is valid. The second is to monitor OPC90 server status. OPC clients utilize the OPC GetStatus method to read a server's status. This function returns several data items regarding the status of the server. One of those items is the server state which can be used to determine if the server is running or faulted due to communication with all the CIU(s) being bad. It returns 1 when running and CIU communication is good otherwise it can be setup to return 2 (fault) when CIU communication is bad. Also, optionally the GetStatus return code can be configured to return RPC\_E\_FAULT (2,147,549,444) when communication with all CIU(s) is bad.

The settings for the GetStatus server state and return code are configured using the Edit | Error Logging dialog.

#### 5.3.11 Error Detection

It is important to understand the two distinct types of log files that OPC90 is capable of generating.

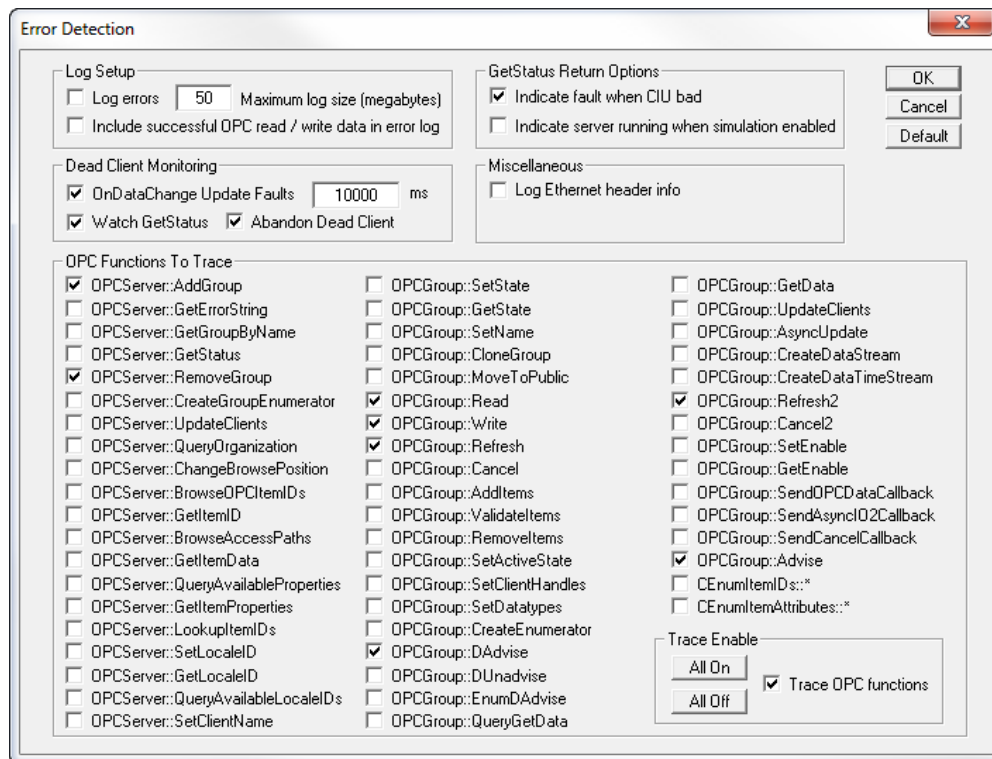
The first type of log is related to diagnosing communication problems with the ABB Bailey CIU device. This log type is enabled by right clicking on the device block within the OPC90 program window and enabling the type of information to log.



In the above example, logging of general CIU errors along with data associated with communication with the CIU has been requested to be logged. The logs are text files viewable by notepad and can be found in the C:\Program Files\OPC90 Server\Log directory. These log file names begin with OPC90 and include the device name along with the time and date the log was generated. The file name “OPC90 CIU Log Mon Jan 10 14.40.52 2011.log” is an example of this log type. It is important to remember that enabling / disabling logs of this type become permanent whenever the database is saved. So for example to log all of the CIU communication on OPC90 startup, enable those logs, save the database and restart OPC90. Afterwards those logs can be turned off and the database saved again so the next time OPC90 is restarted, the logging of the startup information will not occur. Note that restarting OPC90 is not required to enable and disable these logs. The example given was the procedure to log communication on startup of OPC90.

The second type of log file is useful when trying to isolate data exchange problems with OPC client software. This log file is enabled / disabled using the OPC90 “Edit | Error Detection” menu item. These logs are text files viewable by notepad and are found in the C:\Program Files\OPC90 Server\Log directory. The log file name begins with OPC90 Error Detection followed by the time and date the log was generated. The file name “OPC90 Error Detection Log Sun Jan 09 17.27.03 2011.log” is an example of this log type.

Select “Edit | Error Detection” to configuration the error detection log options. These settings are retained in the Windows registry so saving the database is not necessary. When this menu item is selected the following dialog is displayed:



When the error detection log is enabled, OPC90 logs all errors returned to OPC clients. It will also log all fatal errors encountered such as CIU communication faults. The maximum size of the log file can be set in the range of 1 to 100 megabytes. When the log file reaches the maximum size it is closed and a new one opened. This is also true of the first category of logs previously mentioned. Error logs (both categories) are automatically deleted when they become older than the number of days configured in the device property. The default setting is 7 days.

The response of the OPC GetStatus function is configurable from this dialog. It can be enabled to return a failed server status when communication with all CIUs is bad. While running in simulation mode the running in test mode status is normally returned. This can be changed to return server running instead should the OPC client(s) have problems with the test mode status.

The returned server status is often used by OPC clients to take corrective actions such as connecting with an alternative OPC server if a faulty status is received. OPC90 returns the following possible statuses.

OPC\_STATUS\_RUNNING is the normal expected return status. This will be returned even when communication with the DCS is bad unless the “indicate fault when CIU bad” option is enabled. If this option is enabled and CIU communication is completely lost, the return status will indicate failure.

OPC\_STATUS\_FAILED is returned when the “indicate fault when CIU bad” option is enabled and CIU communication is completely lost.

OPC\_STATUS\_NOCONFIG is returned when the database has not been setup yet or something is preventing OPC90 from loading its database.

OPC\_STATUS\_TEST is returned when OPC90 simulation is enabled. This status is an indication the data being received is not live information (it's being simulated) and is of a test nature. The "indicate server running when simulation enabled" option can be set to request OPC\_STATUS\_RUNNING to be returned when simulation is enabled.

Miscellaneous section allows Ethernet header information to be included when logging CIU messages associated with IET800 style CIU.

Logged messages include the time of the log (millisecond resolution) along with the OPC client number (or name if the OPC client sets its name) for which the error occurred. In addition to OPC errors, successful read and write OPC operations (enabled in the first category of logs previously mentioned) can be included in the log. This is useful for constructing sequence of events between normal operation and when errors occur. Finally, the ability to include OPC function tracing can be enabled for all, or specific functions supported by OPC90. OPC function tracing can be used to log the time when a function is called and the time when it returns to the OPC client. This is a useful feature when trying to track down problems related to DCOM and network communication.

OPC90 includes logic that can attempt to detect dead OPC clients. If an OPC client faults during operation, it obviously can't perform a graceful disconnect from OPC90 wasting memory resources. The dead OPC client monitoring section can be used to enable two types of dead OPC client detection logic.

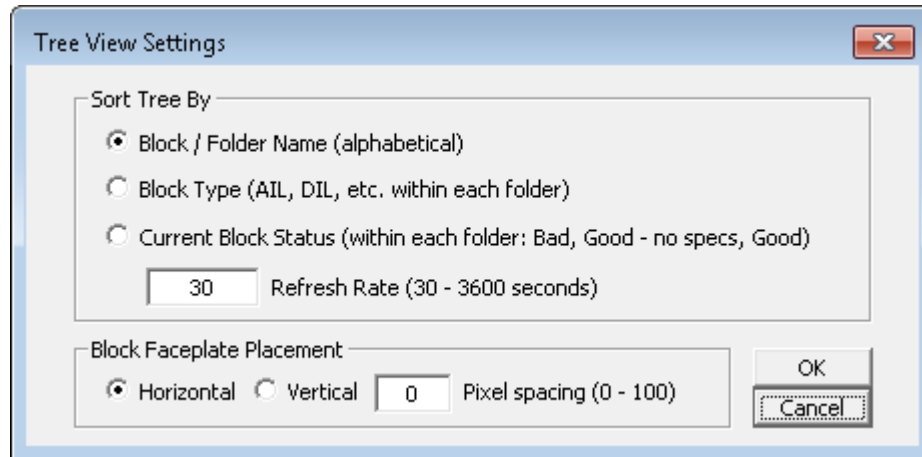
The first is faults that occur when OPC90 tries to send OnDataChange messages to the dead OPC client. Error responses to these updates are an indicator that the OPC client has faulted.

The second is monitoring for the absence of GetStatus messages that OPC clients typically send to servers. Do not set this option if there are a mixture of OPC clients on the same PC running OPC90 and external PCs. It is ok to set if all OPC clients are on the same PC as OPC90 or all OPC clients are on external PCs to the one running OPC90.

Setting "abandon dead client" causes OPC90 to give up memory associated with a declared dead OPC client as detected by either of the two dead OPC client monitoring options.

#### 5.3.12 Tree View Settings

Select "Edit | Tree View Settings" to select how the database tree is sorted and how multiple block faceplates are positioned. The following dialog is displayed:



Three different sorts are available. The first is alphabetical by block and folder names. This sort is useful for flat databases (all blocks configured under the device block). Databases having folders will have the blocks alphabetized within each folder.

The second sort is by block type. All blocks of a given type are shown alphabetically within that type. This sort is applied within each folder for non-flat databases.

The third sort is by block status. This is useful for finding blocks that aren't configured correctly and aren't receiving data. They will appear at the beginning of the database list. As with the other sort types it applies within each folder. This sort has a refresh time associated with it. The database tree is updated at the refresh rate. A refresh can be selected anytime using the View | Refresh menu selection.

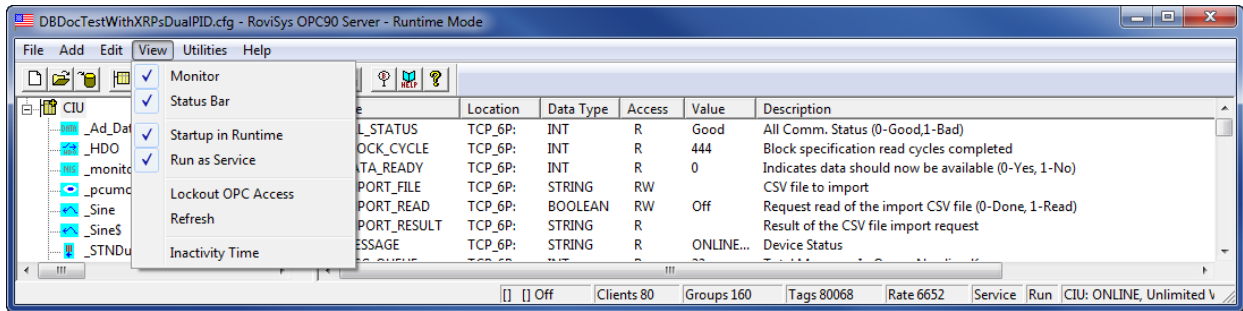
Blocks shown as red indicate they haven't received data yet or have bad quality. Bad quality can be for several reasons such as bad block type, bad block address, field input is bad, node is offline, module is in error or configure mode. Blocks will be yellow when they have received data but still waiting to receive its specification data (i.e. alarm limits, ranges, etc.). Blue blocks mean everything is good. The block ".MESSAGE" tag provides possible explanation of the error condition for blocks that don't turn blue.

While in monitor mode, when a block is double clicked, it's faceplate will be displayed. Placement of multiple block faceplate can be horizontal or vertical. Spacing between each faceplate can also be specified, expressed in pixels.

## 5.4 View

The view menu option allows selection of monitor mode, status bar, startup in runtime and run as service options. The following picture shows these available View menu items:

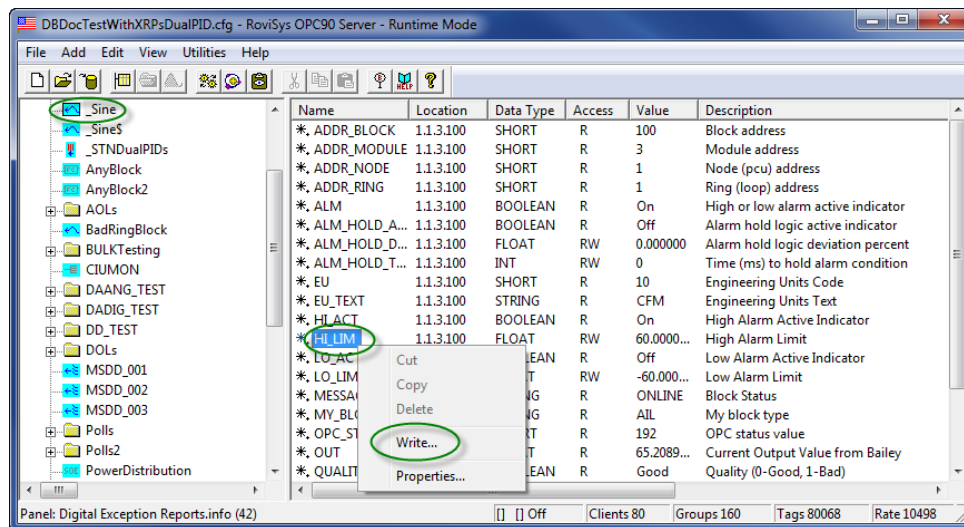




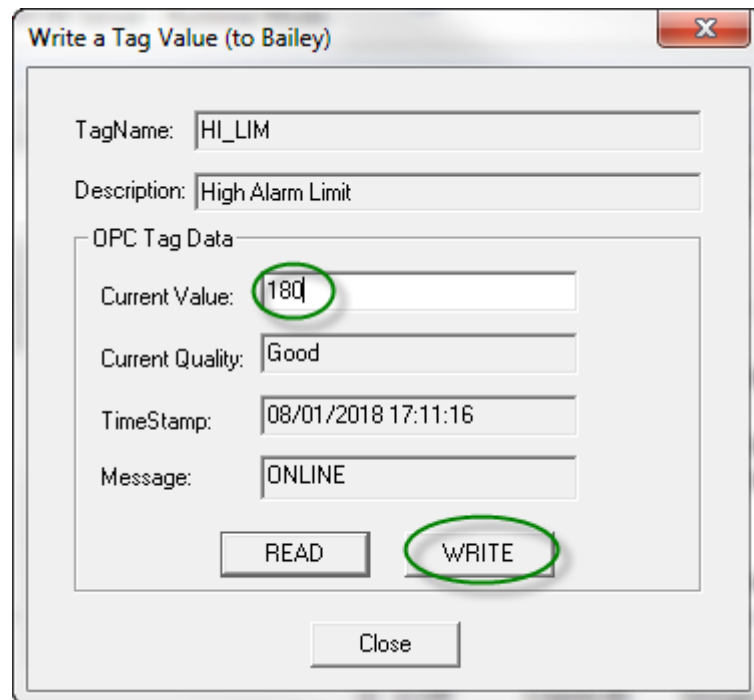
#### 5.4.1 View Monitor

Select “Monitor” to request that OPC90 go into the monitor mode of operation. In this mode the real time values being received from Bailey are displayed in the program window. Monitor mode is useful for instructing OPC90 to begin collecting data from the ABB Bailey system even when no OPC clients are connected. This feature allows the OPC90 configuration to be validated prior to developing any OPC client connections.

The following picture shows monitor mode of operation viewing the tag values of a block named `_Sine`:

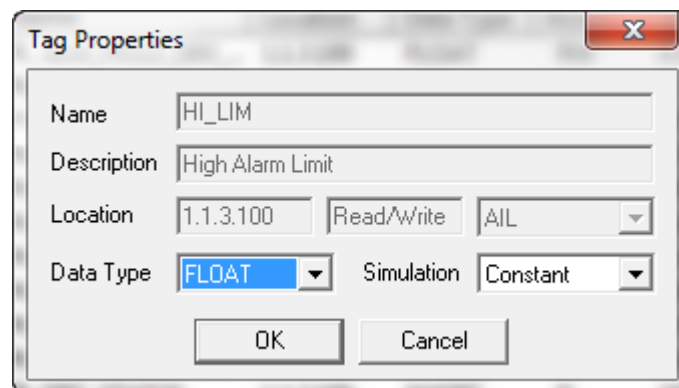


Some tag values have write access and can be written from within the OPC90 program window when monitor mode is active. Simply right click on the tag name and select the write option. The following dialog will be displayed:



Enter the desired value to be written in the current value field and click on the write button. The entered value will be written.

The tag properties can also be modified. Right click on the tag name and select the properties option. The following dialog will be displayed:



This dialog allows the default native data type representation to be changed. When simulation is enabled, the type of signal to be applied can also be specified. Note that changes to the data type and simulation signal apply globally to all blocks of the same type. In other words if the native type or simulation signal for the OUT tag is changed for the AIL block, the change applies to all AIL blocks.

#### 5.4.2 View Status

Select "Status Bar" to enable and disable the program status bar. When OPC90 is not in runtime mode, the status bar presents tag summary information about the last selected

database entity. When in monitor mode it presents the operational state of the selected database entity.

The status bar also presents the state of database shadowing (off, on), current number of attached OPC clients, total groups and tags those OPC clients have setup along with the current OPC data exchange rate for those tags. A field shows if OPC90 is running as a service, process started by a OPC client or the user. The run time state (run / stopped) is also displayed.

#### 5.4.3 View Start In Runtime

Select “Start In Runtime” to enable or disable the mode of operation in which OPC90 automatically starts its communication logic when the program is first executed. This occurs regardless of whether or not any OPC clients are attached to it. Start in runtime is useful when OPC90 is executed automatically as part of a login startup group. It causes the database to be established with the Bailey interface and be made ready for access prior to any OPC client attaching to OPC90. It is most useful for large databases that could take minutes to complete the Bailey interface startup logic. Note that if startup in runtime is enabled and monitor mode enabled and then disabled when there are no outstanding OPC clients attached to OPC90, the runtime mode of operation is stopped. The program status bar indicates the state of runtime. It indicates “Run” when currently enabled and “Stop” when not currently active.

#### 5.4.4 View Run As Service

Select “Run As Service” to enable or disable OPC90 as a Windows service. A dialog will be displayed confirming whether or not the service should be installed or deleted. Running as a service is recommended for Windows 2000 / XP / 2003 Server / 7 / 10 / 2008 / 2012 / 2016 Server systems.

When run as a service is enabled, it is initially setup to run from the system account. This setting is valid when OPC90 and the OPC client(s) are intended to run on the same PC and both run within the local system account. If remote OPC client(s) need to access OPC90 running as a service it should be changed to run in a specific privileged account. The same is true if the configuration shadowing feature is to be enabled. Use the Windows service manager to change the OPC90 service to run in that specific account. If using a workgroup, the remote OPC client(s) must also be setup to run within the same account name and password on their respective PC(s).

OPC90 has special logic that allows a second instance of the program to detect the service is running and attaches a user interface to the service. Therefore, allowing it to interact with the desktop is not required. Note that this special logic is supported for Windows XP / 2003 Server / 7 / 10 / 2008 / 2012 Server / 2016 Server.

Running OPC90 as a service along with the “Start In Runtime” and “Monitor” option is ideal for automatic system startup after a PC reboot and continuous runtime mode even after all OPC clients detach from the server. On startup, OPC90 will establish its database with the Bailey system prior to any OPC clients attaching to it, reducing the

time it takes to gain initial access to database points. After all OPC clients attach and detach, runtime will not be stopped if monitor is still enabled. If monitor mode is not enabled and all OPC clients detach, runtime will be stopped. As soon as one or more OPC clients attach, runtime is automatically started again.

Note that it is not absolutely necessary to run OPC90 as a service since OPC by definition automatically starts the server when any OPC client makes a request for its data. Therefore, if problems are encountered with OPC clients attaching to OPC90 running as a service that cannot be resolved by also running the OPC client as a service, or within the same account the OPC90 service is running, then running as a service should be disabled.

#### 5.4.5 View Lockout OPC Access

This selection is a toggle which enabled / disables the Lockout OPC Access feature. When OPC access is in the lockout state, OPC90 rejects all attempts by OPC clients to attach with it.

This feature is useful when attempting to open a different database. Opening a database is not allowed when OPC clients are attached. In some situations it is difficult to stop all OPC clients so this lockout feature can be enabled to stop the OPC clients from making a connection. It can also be enabled prior to running a QuickUpdater when the QuickUpdater fails because the OPC clients are restarting OPC90 before it can be updated.

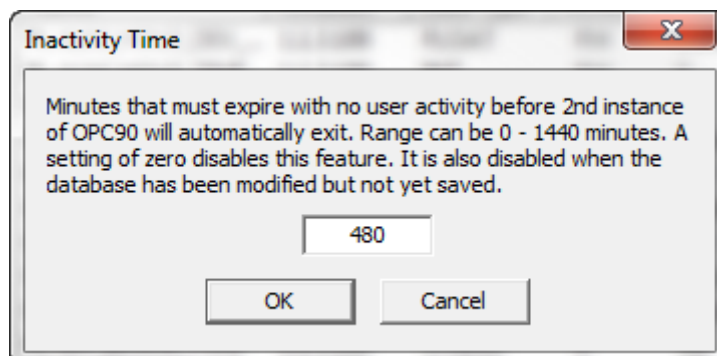
Do not leave this feature enabled since it defeats the purpose of what OPC90 is designed to do. It is included as a setup and maintenance feature. If accidentally left on, each time OPC90 is started the user will be prompted to turn it off.

#### 5.4.6 View Refresh

This option causes the database tree list to redraw with the latest information. It is useful after adding or modifying block names. When the tree view setting is sort by status this is useful for getting the most current status sort before the refresh period occurs.

#### 5.4.7 View Inactivity Time

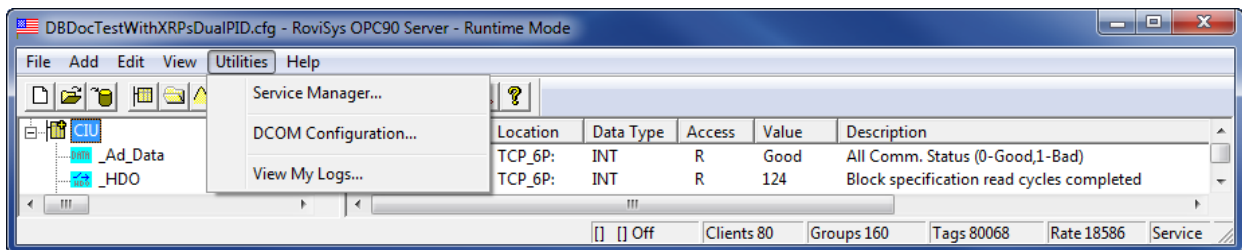
This selection displays the following dialog which allows definition of the inactivity time.



This feature is intended to reduce unnecessary processing requests on the OPC90 service. It is especially useful in terminal service environments when multiple users are logged in monitoring OPC90 server and they forget to exit the 2<sup>nd</sup> instance when they are done using it.

## 5.5 Utilities

The utilities menu option can be used to easily run the Windows service manager or DCOMCNFG program. It can also be used to view OPC90 device and error detection log files. The following picture shows these available View menu items:



### 5.5.1 Service Manager

Selecting the Utilities | Service Manager menu option results in the Windows Service Manager to run. That program allows the OPC90Server service properties to be modified and the service started. The service manager is automatically run by OPC90 when it has been requested to set itself up as a service (see View | Run as Service).

### 5.5.2 DCOM Configuration

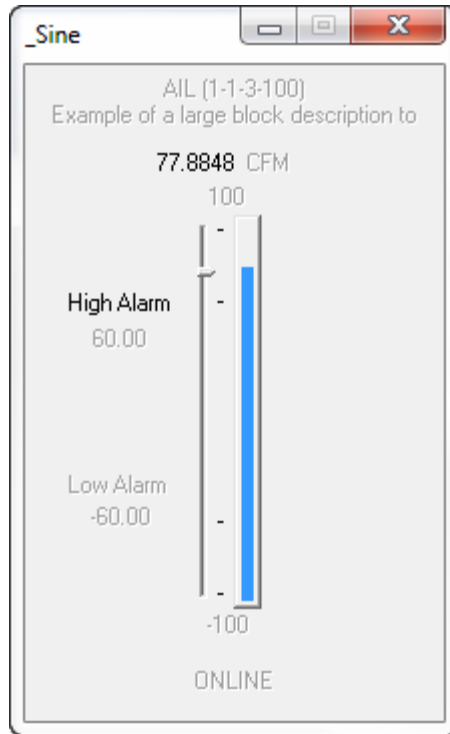
Selecting the Utilities | DCOM Configuration menu option results in the Windows DCOMCNFG program to run. That program allows DCOM to be setup for the computer and OPC90.

### 5.5.3 View My Logs

Selecting the Utilities | View My Logs menu option displays a file open dialog allowing a specific OPC90 device or error log file to be selected and viewed by opening it in notepad.

## 5.6 Block Faceplates

Block faceplates are a handy engineering feature that allows real time monitoring and control of any block in the database. The faceplates are available whenever OPC clients are attached to OPC90, the startup in runtime option is enabled or it is in monitor mode. Double click on the block to display its faceplate. Another method is to right click on the block and select Faceplate | Display. The faceplate dialog will be displayed that corresponds to the block type selected. The following example faceplate corresponds to the AIL block type.



All faceplates have a title, which is the block name. The top two lines in the faceplate display the block type and address in ring-node-module-block format. The block description appears immediately under this information. Along the bottom is the block operational status. The remaining indicators and controls that appear will be specific to the block type.

Some faceplates allow numbers to be entered. These types of entries should be completed by pressing the “Tab” key although the “Enter” key also works for all blocks except the Device block. If an entry is not completed within a few seconds, it will revert back to the original setting. Many blocks have alarm and other operational indicators. These indicators are in the inactive state when gray and active when black. For this example, the high alarm is active.

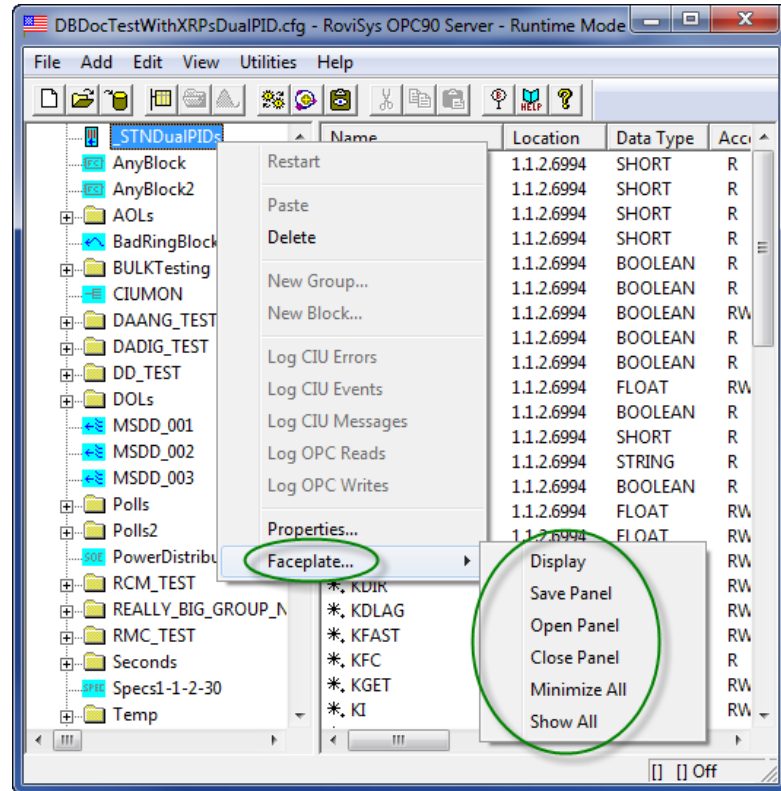
It is important to note that all faceplates have been designed to only utilize OPC tags available from its corresponding block type. This means the faceplates for each block type can serve as a good example on how a faceplate might be implemented in any given OPC client accessing the block data.

### 5.6.1 Faceplate Panels

The ability to display multiple faceplates was recently added to OPC90. This is a handy feature for monitoring many values associated within a specific area in the plant. The blocks containing the desired data to view are double clicked to show their faceplate. As each block is clicked the faceplate is displayed next to the previous faceplate horizontally or vertically. See Edit | Tree View Settings to select the desired placement. Since different

types of blocks have different size faceplates if there is any overlap, they can be manually moved.

Once all the needed faceplates are on display, they can be saved in a panel file for later callup. Right click on any node in the OPC90 database tree and selecting Faceplate shows the following selections.



#### 5.6.1.1 Display

Displays the faceplate for currently selected block in the OPC90 database tree.

#### 5.6.1.2 Save Panel

Saves the currently opened faceplates in a panel file. Provide a meaningful file name that describes the area or purpose for the collection of opened faceplates. Panel files are saved in subdirectories under the Panels directory (located in OPC90 program directory). The name of these subdirectories are identical to the name of the currently running database. This allows panels to be saved that correspond with different databases.

There is no limit to the number of panel files or faceplates stored in those files. Panel files can be overwritten but they are never deleted by OPC90. Users can delete them using Explorer if they are deemed no longer required. The OPC90 program windows status bar (bottom left) displays the currently opened panel file with a count of the number of faceplates it contains. Prior to opening a panel file it will show "Panel: None" when building a new one.



#### 5.6.1.3 Open Panel

Opens a previously saved panel file. If a panel file is already opened, it will be closed before opening the user selection. This means the faceplates currently on display (and minimized) are automatically dismissed before displaying the ones defined in the panel file being opened. If a panel file contains a reference to a block that has been deleted, it will be skipped and go onto the next valid one found within the file. This is indicated by a blank spot where the faceplate was originally located. The faceplates can be re-arranged or a new faceplate added in the blank spot as desired. Save the update to the panel file just opened.

#### 5.6.1.4 Close Panel

Closes the currently opened panel file. This means the faceplates currently on display (and minimized) are automatically dismissed. "Panel: None" will be displayed in the OPC90 program window status bar (bottom left).

#### 5.6.1.5 Minimize All

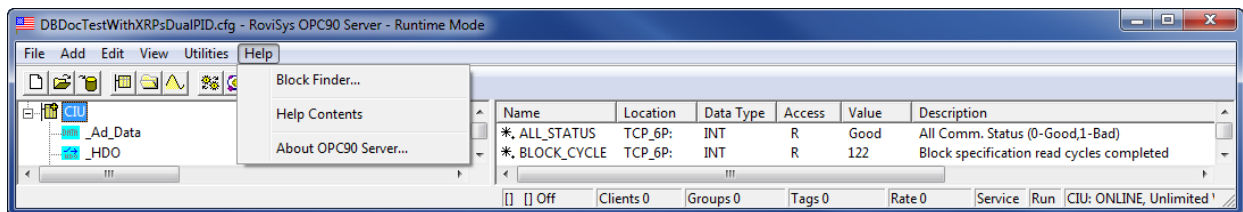
Minimizes all of the currently displayed faceplates. Individual faceplates can be selected for display from the Windows taskbar (bottom of screen) based on their block names.

#### 5.6.1.6 Show All

Shows all the currently minimized faceplates.

## 5.7 Help

The following picture shows the available Help menu items:



Select "About OPC90 Server" to determine the current program version and licensing information.

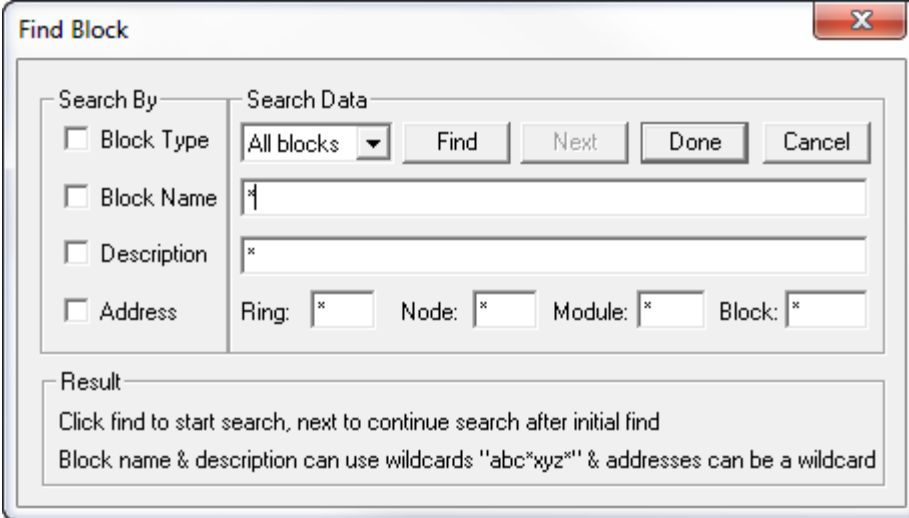
Select "Help Contents" to invoke the OPC90 help file.

Select "Block Finder" to easily search the database for specific blocks.

#### 5.7.1 Block Finder

When "Block Finder" is selected the following dialog is displayed. This dialog is used to easily search the database for specific blocks.





The 'Find Block' dialog box is used to search for specific blocks in the OPC90 system. It features a 'Search By' section with four checkboxes: 'Block Type', 'Block Name', 'Description', and 'Address'. The 'Search Data' section contains a dropdown menu for 'Block Type' (set to 'All blocks'), a 'Find' button, a 'Next' button, a 'Done' button, and a 'Cancel' button. Below these are text input fields for 'Block Name' (containing '\*'), 'Description' (containing '\*'), and 'Address' (containing 'Ring: \* Node: \* Module: \* Block: \*'). A 'Result' section at the bottom provides instructions: 'Click find to start search, next to continue search after initial find' and 'Block name & description can use wildcards "abc\*xyz"' & addresses can be a wildcard'.

A search can be built using the “and” of up to four conditions (block type, block name, description and address). Click on the conditions to be included in the search. Next fill in the search data to be used for each of those conditions. The block type search can be for all block types or a specific block type. Both the block name and description can be for an exact match (not case sensitive) or partial strings using the wildcard character \*. The following wildcard combinations are supported.

\*substring1  
 \*substring1\*  
 substring1\*  
 substring1\*substring2  
 substring1\*substring2\*

The address search condition can also include a wildcard for any of the address fields (ring, node, module and block).

So for example the following setup searches for OPC90 STN blocks with block names starting with “STN” and having “00” in the name one or more characters past “STN” but not at the end of the name. The module address for the OPC90 block must be module 3.

The image shows a 'Find Block' dialog box with a title bar containing a close button (X). The dialog is divided into two main sections: 'Search By' and 'Search Data'.

**Search By:**

- ☒ Block Type
- ☒ Block Name
- ☐ Description
- ☒ Address

**Search Data:**

- Block Type dropdown: STN
- Find button, Next button, Done button, Cancel button
- Block Name text box: \*STN\*
- Description text box: \*
- Address section:
  - Ring: \*
  - Node: \*
  - Module: 2
  - Block: \*

**Result:**

Found: CIU\_StnDualPIDs (STN) @ 1-1-2-6994  
Test STN referencing 2 PIDs

The search starts at the beginning of the database when the find button is clicked. Click the next button to continue the search from the previous block that matched the search condition. When a block is found matching the requested search condition it is shown in the results section. The block name is given prefixed by its full path position. The block type and address is also provided. The block description is beneath the block name. Also the block tree location is automatically selected and opened in the OPC90 main program window.

After searching is complete, exit the dialog by clicking done which saves the search setup data for the next time the searching is invoked. Click on cancel to exit the dialog without saving the search setup data.

## 6 Groups

There are two different types of groups to keep in mind. The first is the OPC90 Server organization of blocks within groups. This is created within the OPC90 Server configuration and is simply a collection of OPC90 Server blocks or other groups. An OPC90 Server group can be compared to a directory within a file system. Where in this case the root directory is always an OPC90 DEVICE block. An OPC90 group is only used to store other OPC90 groups or collections of similar OPC90 blocks. No tags exist within an OPC90 group (tags only exist within an OPC90 block).

When OPC90 is not in runtime operation it supports drag and drop at the group and block level. A block can be moved from one group to another by dragging it to another group and dropping it there. Likewise, dragging an entire group is also supported. All blocks within that group are included in the drag and drop operation. The drag and drop feature is also disabled when a second instance of OPC90 is being used as the user interface when running as a service or embedded (started automatically by an OPC client access).

The other type of group is created by and exists within an OPC client and is referred to as the OPC client group. The OPC client creates this group after it has successfully connected to the OPC90 Server in runtime. The OPC client group is an interface into the server. After an OPC client group is created, OPC tags can then be added to it in runtime. Thus the OPC client group acts as a collection of tags not blocks. OPC90 Server address space can be browsed and then OPC90 Server tags selected in runtime. These tags are then placed into the OPC client group. An OPC client group does not have to match an OPC90 Server group. When the OPC client disconnects from the server all tags are disestablished and the OPC client group does not exist anymore.

## 7 Blocks

The following sub-sections define the OPC90 Server blocks available for interfacing with Bailey systems. For ease of access, the blocks are presented in alphabetical order.

It is important to note that a Device Block (DEVICE) **must** be configured for the OPC90 Server to successfully communicate with the Bailey system. This block declares an instance of the Bailey driver and its operational characteristics. The remaining OPC90 Server blocks provide access to the various Bailey exception report blocks, polling outputs of non-exception reported blocks, system status, configuration reading and tuning of any Bailey block in the system and the output report block types. Note that the output report block types can only be used with the Bailey NCIU02/03/04, INPCI02 and INIClxx interfaces.

Each OPC90 Server block definition includes a table, which summarizes the attributes and tag names supported by the block type. This table contains four columns which are "Attribute / Tag Name", "Data Type", "Access" and "Description".

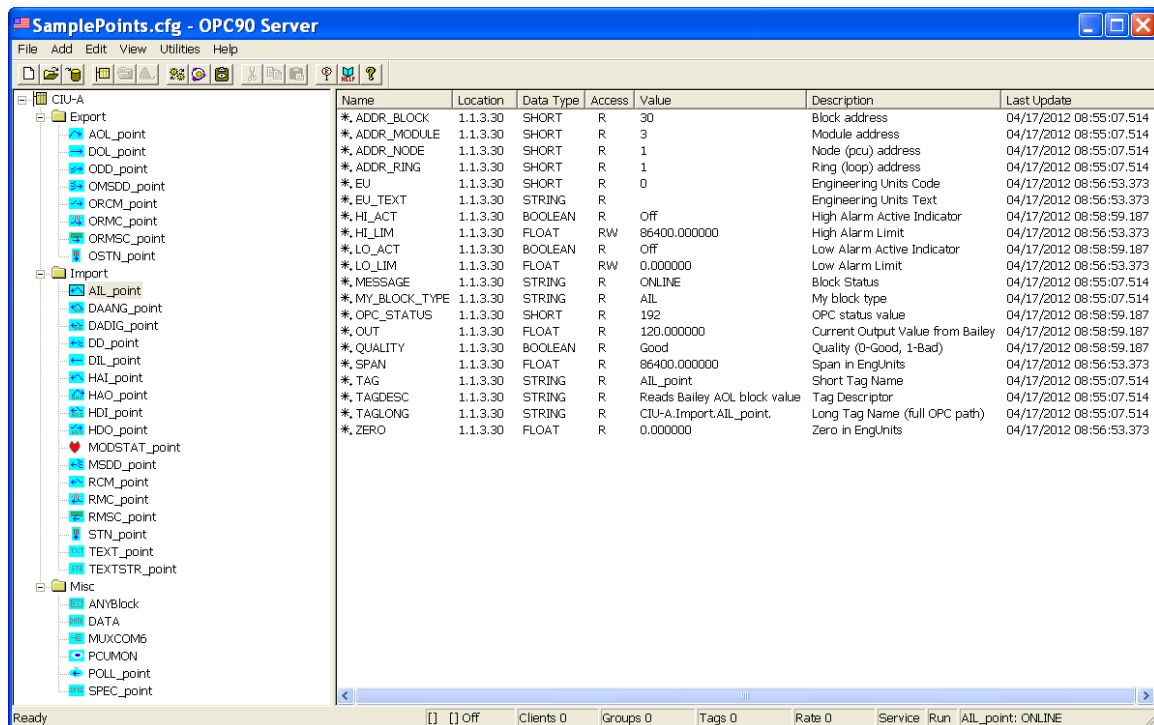
"Attributes" are fields that must be setup as part of defining the block within the OPC90 database. Most of these fields are not available to the OPC client as an OPC tag. OPC90 supports online changes to some of these attributes via the block properties, available by right clicking on the block and selecting properties. This feature is handy for correcting block address definition problems when OPC90 is currently online providing data to OPC clients. Note that the block properties dialog is the same one used to configure new blocks (see preceding add block section). Just enter the correct address, and OPC90 will re-establish the point within the ABB Bailey interface. Once the update has been confirmed to correct the addressing problem, don't forget to save it using the OPC90 File->Save menu selection.

"Tag Names" are fields available to the OPC client. The "Data Type" column gives the default OPC tag type for each defined tag name.



The "Access" column defines how the OPC client may use the tag name. A value of "Config/Read" defines the entry as an attribute (attributes are also identified with *italicized* text). A value of "Read/Write" means the OPC client is expected to provide the value for that tag name. A value of "Read/Write" means the OPC client can both read and write the value which is also sourced by the Bailey system. For this type of tag access, values received from the OPC client will be written to Bailey but OPC90 will also look for values received from the Bailey system and write them to the OPC client via the associated tag name.

The "Description" column gives the purpose for each attribute and tag name.

Included with the OPC90 installation is a sample configuration file called "SamplePoints.cfg". This file contains examples of all blocks supported and can be viewed by selecting OPC90 File->Open. The following picture presents an example of the sample point configuration:



This example shows a device called “CIU-A”. Three groups have been configured under this device called “Export”, “Import” and “Misc”. The “Import” group contains sample blocks for receiving exception reported values from the ABB Bailey system. The “Export” group contains sample blocks for responding to the ABB Bailey system as the indicated exception report block. The “Misc” group contains sample blocks for reading / tuning ABB Bailey block configuration information, multiplexing the COM6 port as a CIU port, polling a block output and providing specification information for a particular block.

As noted from this example, the device has a unique symbol  identifying it as a device. Groups are identified with the folder  symbol. Each block type has a unique symbol identifying the functionality of that block. Generally, blocks that import exception reported values are identified with an arrow pointing left (into the device symbol). Blocks that export exception reported values are identified with an arrow pointing right (out from the device symbol). The definition of each block in the following sub-sections includes the symbol associated with that block.

It is **important** to note that the names given to devices, groups and blocks determine the OPC client tag name used to access OPC data. In the above example the OPC client tag name:

“CIU-A.Import.AIL\_point.OUT”

will access the output value of the block named AIL\_point.

To read and write a set point value of the block named STN\_point, the OPC client tag name for the above example would be:

“CIU-A.Import.STN\_point.SP”

## 7.1 Analog Input Loop (AIL)

The AIL OPC90 block is used to retrieve the exception reported output from Bailey Analog Output / Loop (function code 30), Analog Output / Loop With Deadband (function code 48), Analog Point Definition (function code 70) and Enhanced Analog Point Definition (function code 158) blocks.

Restrictions: This OPC block can be utilized with all Bailey interface types except serial port module (SPM & CPM02).

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey module address.
ADDR_BLOCK	VT_I2	Config/Read	Bailey block address.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
ALM_HOLD_ACT	VT_BOOL	Read	Alarm hold logic active indicator (see note 1).
ALM_HOLD_TIME	VT_I4	Read/Write	Alarm hold time in milliseconds (see note 1).
ALM_HOLD_DEV_PCT	VT_R4	Read/Write	Alarm hold deviation percent (see note 1).
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey values (0-good, 1-bad).
OUT	VT_R4	Read	Current output value from Bailey.
SPAN	VT_R4	Read	Span in Engineering Units.
ZERO	VT_R4	Read	Zero in Engineering Units.
EU	VT_I2	Read	Engineering Units Code.
EU_TEXT	VT_BSTR	Read	Engineering Units String.
HI_LIM	VT_R4	Read/Write	The setting for the alarm limit used to detect the high alarm condition. Writing a new value to this attribute automatically tunes the alarm limit of the associated Bailey block.
LO_LIM	VT_R4	Read/Write	The setting for the alarm limit used to detect the low alarm condition. Writing a new value to this attribute automatically tunes the alarm limit of the associated Bailey block.
HI_ACT	VT_BOOL	Read	High alarm active indicator.
LO_ACT	VT_BOOL	Read	Low alarm active indicator.
ALM	VT_BOOL	Read	High or Low alarm active indicator.

Notes:

- 1.) Alarm hold logic holds the HI\_ACT and LO\_ACT alarm indicators after the alarm condition no longer exists. The ALM\_HOLD\_ACT tag indicates a hold condition is in effect when TRUE (1). The duration of the hold is defined by ALM\_HOLD\_TIME expressed in milliseconds and / or ALM\_HOLD\_DEV\_PCT which defines the deviation expressed as percent of the block SPAN. When the block OUT value deviates from the alarm limit by the calculated deviation amount, the alarm hold is cleared. The ALM\_HOLD\_TIME and ALM\_HOLD\_DEV\_PCT tags are read / write which means they can be changed in real time from the OPC client. Setting them to zero

disables that particular hold feature. The accuracy of ALM\_HOLD\_TIME time is 100 milliseconds.



## 7.2 Analog Output Loop (AOL)

The AOL OPC90 block is used to send an exception reported output generated by updates to the block input tag (IN). *Note that this block does not received data from an AOL block (FC 30) running in a Bailey controller. Use the OPC90 AIL block for that purpose.*

The exception reports generated by this block can be received by a Bailey console analog tag, Bailey Analog Input / Loop blocks (function code 26) and Analog Input / Inifinet blocks (function code 121). The OPC90 AOL block will automatically generate the appropriate quality, and alarming status based on the value presented at the block input.

Restrictions: This OPC block can be utilized with all Bailey interface types except serial port module (SPM, CPM02 & CPM03) and computer interface command series (CIC).

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey CIU ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey CIU node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey CIU module address.
ADDR_BLOCK	VT_I2	Config/Read	Block number to establish this point at within the Bailey interface (see note 1).
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
INIT_VALUE	VT_R4	Config/Read	Initial startup value
ZERO	VT_R4	Config/Read	Zero of IN/OUT value in engineering units code.
SPAN	VT_R4	Config/Read	Span of IN/OUT value in engineering units code.
SIG_CHANGE	VT_R4	Config/Read	Amount that IN must change, cycle to cycle before it will be reported to Bailey. Expressed as a percentage of the SPAN.
EU	VT_I2	Config/Read	Bailey engineering units code to use when establishing this point.
MAX_TIME	VT_I4	Config/Read	If IN never changes significantly, it will be reported at the maximum time interval (seconds) defined by this attribute.
HI_LIM	VT_R4	Config/Read/Write	The setting for the alarm limit used to derive the high alarm condition (see note 2).
LO_LIM	VT_R4	Config/Read/Write	The setting for the alarm limit used to derive the low alarm condition (see note 2).
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
IN	VT_R4	Read/Write	Input value to be exception reported to Bailey.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read/Write	Current quality (see note 3) of Bailey values (0-good, 1-bad).
OUT	VT_R4	Read	Last input value reported to Bailey.
HI_ACT	VT_BOOL	Read	High alarm active indicator.
LO_ACT	VT_BOOL	Read	Low alarm active indicator.

Notes:

- 1.) Make sure the block number attribute is unique with respect to the other AOL, DOL, ODD, OMSDD, ORCM, ORMC, ORMSC and OSTN OPC90 blocks associated with the same Bailey Interface Definition block. The block number attribute must also be defined within the range of 1 to maximum number of allowed outputs set up within the associated OPC90 DEVICE Block. The Bailey system will receive data from this block at the ring and node address of the Bailey CIU interface, module two and block number defined by the block attribute.
- 2.) This attribute is normally configured when the block is first defined. Since it is associated with the alarm limits of value an OPC client is permitted to write new alarm limits in run time. Care should be taken to not continuously change the limit value since each change necessitates dis-establishing the point from the ABB Bailey interface and than re-establishing it with the new limit value.
- 3.) When the Device block "Set bad quality of max exception timeout" property is enabled, the QUALITY tag of this block will be set bad if writes to the block input(s) do not occur within the Exception Report Output Max Time setting of the block. When this occurs, bad quality will also be written to the CIU and therefore propagated to the users of the block data within the ABB Bailey system. The quality can also be set bad by writing a one (1) to this tag.

### 7.3 Block Data (BLK)

The BLK OPC90 block provides Bailey function block configuration capabilities. It supports reading Bailey configuration, tuning specifications, changing module modes, writing new blocks, modifying existing blocks, deleting blocks and saving and restoring module configurations. The BLK block is designed to allow MMI OPC clients to easily set up a Bailey function block programming environment.

Restrictions: None, this OPC90 block can be utilized with all Bailey interface types.

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/ Read/Write	Bailey ring address.
ADDR_NODE	VT_I2	Config/ Read/Write	Bailey node address.
ADDR_MODULE	VT_I2	Config/ Read/Write	Bailey module address.
ADDR_BLOCK	VT_I2	Config/ Read/Write	Bailey block address.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Message indicating result of the last configuration related request.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
READ	VT_BOOL	Read/Write	Set to request the specifications for the addressed block to be read. The driver will reset this attribute when the read request is completed.
NEXT	VT_BOOL	Read/Write	Set to request a read of the specifications for the next block from the current one. The driver will reset this attribute when the next block read request is completed and update the BLOCK attribute to reflect the next block read.
TUNE	VT_BOOL	Read/Write	Set to request a tune operation for any specifications that have been changed after reading them. The driver will reset this attribute when the tune read request is completed.
DEFAULT	VT_BOOL	Read/Write	Set to request a read of the default specification settings for the function code indicted by the FC attribute. The driver will reset this attribute when the read default request is completed.
PASSWORD	VT_BSTR	Read/Write	Password to unlock the OPERATION attribute and some commands supported by the COMMAND attribute. Will indicate "???????" when locked and "*****" when unlocked.
FILE_TYPE	VT_BOOL	Read/Write	When reset the configuration file type used by the save and load command is the RoviSys OPC90 "C90" file type format. When set the ABB Bailey "CFG" file type format is used.
COMMAND	VT_BSTR	Read/Write	Supports a variety of Bailey controller configuration activities using simple text based

			commands. See user configuration command processing for usage details.
COMMAND_ACT	VT_BOOL	Read	Set when a text based configuration command is active, otherwise reset when command is not active or completed.
OPERATION	VT_I2	Read/Write	Commands Bailey controllers to different operating modes per the following values: 1 = request controller software reset 2 = request configure mode 3 = request execute mode 4 = request configuration initialization The driver will reset this attribute to zero when the module operation request has been completed.
MODE	VT_BSTR	Read	Returns the current operating mode (Execute, Configure, Error) of the module addressed by the ADDR_MODULE attribute.
WRITE	VT_BOOL	Read/Write	Set to request a block to be written or modified. The driver will reset this attribute when the block write or modification request is completed.
DELETE	VT_BOOL	Read/Write	Set to request a block to be deleted. The driver will reset this attribute when the delete block request is completed.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey values (0-good, 1-bad).
ACKNAK	VT_I2	Read	Result of the last block configuration activity. A non-zero code indicates an error has occurred.
FC	VT_I2	Read/Write	Function code number returned for the last block read. Also can be written by OPC client when in the process of reading default function code specifications.
FCNAME	VT_BSTR	Read	Function code name returned for the last block read or default specification read.
SX_COUNT	VT_I2	Read	Number of valid specifications for last block Read/Write.
SPEC_FORMAT	VT_I2	Read/Write	Determines how specification description data is to be formatted (see note 1).
S01 to S63	VT_BSTR	Read	Descriptions for specifications 1 through 63.
S01_STRING to S63_STRING	VT_BSTR	Read/Write	Value for specifications 1 through 63 represented as a string.
S01_TYPE to S63_TYPE	VTI2	Read	Type for specifications 1 through 63 (see note 2).
S01_VALUE to S63_VALUE	VT_R4	Read/Write	Value for specifications 1 through 63.

Notes:

- 1.) The format of how the specification data is returned can be selected utilizing the SPEC\_FORMAT attribute. The following choices are available:
  - 0 - Sx Name Value Format includes spec number, its name and current value
  - 1 - Name Value Format includes spec name and its current value

- 2 - Sx Name                      Format includes spec number, and its name
- 3 - Name                         Format includes just the name of the spec

2.) The specification type is defined according to the following values. Tunable specifications will be indicated by 100 added to these values.

- 1 = REAL\_2
- 2 = REAL\_3
- 3 = BOOLEAN
- 4 = INTEGER\_1
- 5 = INTEGER\_2
- 7 = INTEGER\_4
- 6 = REAL\_4
- 8 = STRING

### 7.3.1 Reading a Bailey Block

To retrieve existing block specification information, the block address must first be written to the ADDR\_RING, ADDR\_NODE, ADDR\_MODULE and ADDR\_BLOCK tags of this block. Once the address is written, the READ tag can be set TRUE which instructs OPC90 to retrieve the block specifications. The specification values are copied to the S01\_VALUE through S63\_VALUE tags. The S01 through S63 tags are also updated with a description of each returned specification. The FC tag will return the block function code number and the FCNAME tag will return the name of the function code. Completion of the read request is signaled when the READ tag is reset to FALSE. The MESSAGE tag will also return a text message indicating completion of the read operation or the reason for failure if the read cannot be completed. When the ACKNAK tag is zero the request was successful, non-zero indicates an error occurred.

Since Bailey function blocks can have up to 63 specifications, tags exist for the maximum number of specifications. The actual number of valid specifications for any given block read can be determined by examining SX\_COUNT tag or the descriptions assigned to each specification (S01 through S63 tags). Null descriptors indicate the specification number is not valid for the block read.

### 7.3.2 Tuning a Bailey Block

Some Bailey specifications can be modified while the block is executing. These are called tunable specifications and are indicated as such by the specification description having the 'T' character in front of it. The tags S01\_VALUE through S63\_VALUE can be used to change the value of tunable specifications. When the TUNE tag is set TRUE, any tunable S01\_VALUE through S63\_VALUE tag that was changed by the OPC client will be included in the Bailey block tune operation. Completion of the tune operation is flagged when the TUNE tag is reset to FALSE. The MESSAGE tag will also return a text message indicating completion of the tune operation or the reason for failure if the tune cannot be completed. When the ACKNAK tag is zero the request was successful, non-zero indicates an error occurred.

### 7.3.3 Reading the Entire Bailey Configuration

The entire Bailey configuration can be read using the READ and NEXT tags (also see save command processing). First start by using the ADDR\_RING, ADDR\_NODE,

ADDR\_MODULE, ADDR\_BLOCK and READ tags to read block zero which is the start of any given module configuration. Each time the NEXT tag is set TRUE, the next block from the last one just read will be retrieved. The returned specification values are copied to the S01\_VALUE through S63\_VALUE tags. The S01 through S63 tags are also updated with a description of each returned specification. The FC tag will return the block function code number and the FCNAME tag will return the name of the function code. Completion of the next read operation is flagged when the NEXT tag is reset to FALSE. The MESSAGE tag will also return a text message indicating completion of the next read operation or the reason for failure if the next read cannot be completed. When the ACKNAK tag is zero the request was successful, non-zero indicates an error occurred. A search for specific function codes could be implemented by monitoring the FC tag while “stepping” through a Bailey configuration.

#### 7.3.4 Changing Bailey Controller Module Modes

Bailey controller modules are always in one of three operating modes. These modes are execute, configure and error. When in the execute mode the controller runs its block configuration. In this mode of operation, existing blocks can be tuned (see section 7.3.2) but new blocks cannot be added and existing blocks cannot be modified or deleted. When in the configure mode, the controller is not running its block configuration. In this mode of operation, existing blocks can be modified or deleted and new ones can be added. A controller module enters the error mode of operation when it detects a configuration error on transition to the execute mode. A controller can also enter the error mode when a trip block (Bailey Function Code 32) receives a trip signal.

The OPERATION tag can be used to change the current operating mode of any Bailey controller. The MODE tag can be used to monitor the module’s current mode of operation. It will return a text message indicating the current operating mode. **WARNING**, changing the operating mode of a controller from execute to configure can cause a plant trip or major outage to occur. For this reason, writing to the OPERATION tag has been placed under password control. When the OPC90 BLK is configured, a case sensitive password is assigned to it. Before OPC90 will accept writes to the OPERATION tag, the OPC client must first write the correct password to the PASSWORD tag. The PASSWORD tag returns the text “????????” when the password has never been received or the received password is incorrect. The PASSWORD tag returns the text “\*\*\*\*\*” when the correct password has been received. After password validation, writes to OPERATION will cause the currently addressed module ( set using the ADDR\_RING, ADDR\_NODE and ADDR\_MODULE tags) to transfer to the requested operating mode. The following OPERATION tag writes are supported:

- 1 = Request module to perform a software reset
- 2 = Request module to enter the configure mode of operation
- 3 = Request module to enter the execute mode of operation
- 4 = Request module to initialize (clear) its current block configuration

Completion of the requested module operation is flagged when the OPERATION tag is reset to a value of zero (0). The MESSAGE tag will also return a text message indicating

completion of the module operation or the reason for failure if the requested operating mode cannot be completed. When the ACKNAK tag is zero the request was successful, non-zero indicates an error occurred. If for example, a request to enter the execute mode actually results in error mode, the MESSAGE tag will display a message indicating why this occurred and the Bailey block that caused the error.

For security, it is highly recommended that once an OPC client has completed its use of the OPERATION tag, it should lock it by sending an invalid password to the PASSWORD tag. Note that OPC90 will automatically invalidate the PASSWORD tag after 60 minutes transpires with no OPC client write activity to the OPC90 BLK block.

### 7.3.5 Writing Bailey Controller Blocks

To write Bailey function blocks to a controller it must be in the configure mode of operation. See the preceding section to understand how to change Bailey controller modes of operation. There are five easy steps an OPC client must follow to write a Bailey function block. First it writes the function code number of the intended new block to the FC tag. Second it writes the new block number to the ADDR\_BLOCK tag. Third it requests the default specification settings for the function code by setting the DEFAULT tag to TRUE. The number of specifications is returned in the SPEC\_COUNT tag, the default specification values in the S01\_VALUE through S63\_VALUE tags and the specification descriptors in the S01 through S63 tags. Completion of the read default request is signaled when the DEFAULT tag is reset to FALSE. The MESSAGE tag will also return a text message indicating completion of the read default operation or the reason for failure if it cannot be completed. When the ACKNAK tag is zero the request was successful, non-zero indicates an error occurred. The fourth step is to change the appropriate specifications from their default values to values that are pertinent to the new block about to be written. This is accomplished by writing to the S01\_VALUE through S63\_VALUE tags. Note that these tags are always written as floating point values. OPC90 will automatically take care of translating the values to the appropriate types required by the Bailey function block. The fifth and last step is to request OPC90 to actually write the block to the Bailey controller. This is accomplished when the WRITE tag is set TRUE. The value of the S01\_VALUE through S63\_VALUE tags appropriate for the block function code are translated to the correct data types required for the Bailey function code and the block write occurs. Completion of the write operation is flagged when the WRITE tag is reset to FALSE. The MESSAGE tag will also return a text message indicating completion of the write operation or the reason for failure if the write cannot be completed. When the ACKNAK tag is zero the request was successful, non-zero indicates an error occurred.

### 7.3.6 Modifying Bailey Controller Blocks

To modify a Bailey function block, the controller must be in the configure mode of operation. See section 7.3.4 to understand how to change Bailey controller modes of operation. There are three easy steps an OPC client must follow to modify a Bailey function block. First it must read the function block to be modified. See section 7.3.1 to understand how to read a block. The second step is to change the appropriate specifications from their current values to the values to be modified. This is accomplished

by writing to the S01\_VALUE through S63\_VALUE tags. Note that these tags are always written as floating point values. OPC90 will automatically take care of translating the values to the appropriate types required by the Bailey function block. The third and last step is to request the block modification by setting the WRITE tag TRUE. The value of the S01\_VALUE through S63\_VALUE tags appropriate for the block function code are translated to the correct data types required for the Bailey function code and the block modification occurs. Completion of the modification operation is flagged when the WRITE tag is reset to FALSE. The MESSAGE tag will also return a text message indicating completion of the modification operation or the reason for failure if the modification cannot be completed. When the ACKNAK tag is zero the request was successful, non-zero indicates an error occurred.

### 7.3.7 Deleting Bailey Controller Blocks

To delete a Bailey function block, the controller must be in the configure mode of operation. See section 7.3.4 to understand how to change Bailey controller modes of operation. There are two easy steps an OPC client must follow to delete a Bailey function block. First it must read the function block to be deleted. See section 7.3.1 to understand how to read a block. The second step is to request the block deletion by setting the DELETE tag to a value of TRUE. Completion of the delete is flagged when the DELETE tag is reset to FALSE. The MESSAGE tag will also return a text message indicating completion of the delete operation or the reason for failure if the deletion cannot be completed. When the ACKNAK tag is zero the request was successful, non-zero indicates an error occurred.

### 7.3.8 Configuration Command Processing

The COMMAND tag allows the OPC client to perform a variety of Bailey configuration related activities using simple text commands. The OPC client sends the text command to the COMMAND tag. It can monitor completion of the requested command using the COMMAND\_ACT tag. The MESSAGE tag will contain a text message showing the result of the command when it has finished. The following table presents an overview of the supported commands:

Command	Arguments	Description
<i>OPERATION</i>	RESET or 1 or CONFIGURE or 2 or EXECUTE or 3 or INITIALIZE or 4 <RING> <NODE> <MODULE>	Changes a Bailey controller mode of operation.
SAVE	<FILE_NAME> <RING> <NODE> <MODULE>	Saves a Bailey controller configuration to a file.
LOAD	<FILE_NAME> <RING> <NODE> <MODULE>	Restores a Bailey controller configuration from a file.
CANCEL	none	Cancels the currently active command.
READ	<BLOCK> <RING> <NODE> <MODULE>	Reads a function block configured in a Bailey controller.
GET	<FUNCTION_CODE> <BLOCK> <RING> <NODE> <MODULE>	Gets the default specifications for a Bailey controller function code.
<i>WRITE</i>	<START_BLOCK> <NUMBER_BLOCKS> <BLOCK_INCREMENT> <RING>	Writes a function block to a Bailey controller.



	<NODE> <MODULE> <SX INCREMENT>	
<i>MODIFY</i>	<START_BLOCK> <NUMBER_BLOCKS> <BLOCK_INCREMENT> <RING> <NODE> <MODULE> <SX INCREMENT>	Modifies an existing function block within a Bailey controller.
<i>DELETE</i>	<START_BLOCK> <END_BLOCK> <RING> <NODE> <MODULE>	Deletes one or more blocks within a Bailey controller.

The command syntax is simple. The commands and arguments are case insensitive. Each argument must be separated by one or more spaces (not commas). Note that the < > characters are not part of the above arguments. They indicate the argument is optional. If an optional argument is omitted the arguments following it cannot be included in the command. The only exceptions to this rule are the WRITE and MODIFY commands. The <SX INCREMENT> argument (where X = specification number and INCREMENT is the amount to add to the specification value for each new block addition) can be added at the end of the command regardless of how many of the other optional arguments are omitted. Multiple <SX INCREMENT> arguments per command are supported.

Note that the italicized commands are under password control. **WARNING**, execution of these commands can cause a plant trip or major outage to occur. They require the correct password to be sent to the PASSWORD tag before the command can be executed. When the OPC90 BLK is configured, a case sensitive password is assigned to it. Before OPC90 will accept protected command writes to the COMMAND tag, the OPC client must first write the correct password to the PASSWORD tag. The PASSWORD tag returns the text "???????" when the password has never been received or the received password is incorrect. The PASSWORD tag returns the text "\*\*\*\*\*" when the correct password has been received. For security, it is highly recommended that once an OPC client has completed its use of the COMMAND tag, it should lock it by sending an invalid password to the PASSWORD tag. Note that OPC90 will automatically invalidate the PASSWORD tag after 60 minutes transpires with no OPC client write activity to the OPC90 BLK block.

### Operation Command

Use this command to change the current operating mode of a controller. See section 7.3.2 entitled "Changing Bailey Controller Module Modes" for a detailed discussion on these modes. The command argument can be the name of the mode or its corresponding mode number. If the address arguments are omitted the current address specified in the ADDR\_RING, ADDR\_NODE, ADDR\_MODULE tags will be used as the destination address for the operation. Following are some example operation commands:

Command	Action
OPERATION CONFIGURE	Changes currently addressed module to configure mode.
OPERATION CONFIGURE 1 3 6	Changes ring 1 node 3 module 6 to configure mode.
OPERATION 2	Changes currently addressed module to configure mode.

OPERATION 2 1 3 7	Changes ring 1 node 3 module 7 to configure mode.
OPERATION EXECUTE 1 3 7	Changes ring 1 node 3 module 7 to execute mode.
OPERATION INITIALIZE 1 3 6	Initializes the configuration of ring 1 node 3 module 6.

### Save Command

Use this command to save a Bailey module function block configuration to a file. Note that only the function blocks are saved. BASIC programs or C programs with any associated data files are not included in the save. Contact RoviSys if your needs include saving programs. Any file name can be specified. If the filename contains spaces it must be enclosed in double quotes. If the file name is excluded, OPC90 will automatically name the file based on the current address specified in the ADDR\_RING, ADDR\_NODE and ADDR\_MODULE tags. For example assume these tags are currently set to the corresponding values of 1, 2 and 3. The file name will be "00100203.C90" or "10203.CFG". This convention makes it easy to identify what Bailey module the file corresponds with. OPC90 can save the configuration in one of two file formats. The FILE\_TYPE tag selects the file format to utilize. When FILE\_TYPE is reset (0), OPC90 will save the Bailey function block configuration data in files that have a "C90" extension. This is the OPC90 file format. When the file name is included in the command, OPC90 will automatically append the "C90" extension if it is missing. Including the "C90" extension in the file name will override the current FILE\_TYPE tag setting. The "C90" files are stored in the "ABB" subdirectory unless an alternative directory path is included as part of the file name. When FILE\_TYPE is set (1), OPC90 will save the Bailey function block configuration data into files that have a "CFG" extension. These files must already exist and are of the format generated by the ABB Bailey CADEWS software and its derivatives. When the file name is included in the command, OPC90 will automatically append the "CFG" extension if it is missing. Including the "CFG" extension in the file name will override the current FILE\_TYPE tag setting. The "CFG" files are also stored in the "ABB" subdirectory unless an alternative directory path is included as part of the file name. If the save command is cancelled or it fails, the file will not be generated and any original file of that name that had been previously saved will remain unchanged. Saving a configuration also generates a second file that documents all of the blocks just saved. This list includes the block number, function code type and specification settings. The name of this file is the saved file name with "\_BLOCK\_LISTING.CSV" appended to it. Microsoft Excel can be used to open the file and look at the blocks contained by the configuration. Following are some example save commands:

Command	Action
SAVE	Saves currently addressed module.
SAVE MFP01	Saves currently addressed module to a file called "MFP01.C90".
SAVE MFP01.CFG	Saves currently addressed module to a file called "MFP01.CFG".
SAVE DIGESTER 1 3 5	Saves ring 1 node 3 module 5 to a file called "DIGESTER.C90".

### Load Command

Use this command to load a Bailey module function block configuration from a previously saved file. Remember the module must first be in configure mode before it can be loaded with a new configuration. It should also be initialized before loading the new configuration (see OPERATION command). BASIC programs or C programs with any associated data

files are not included in the load. Contact RoviSys if your needs include loading of programs. Any file name can be specified. If the file name includes spaces it must be enclosed in double quotes. If the file name is excluded, OPC90 will automatically load a file name based on the current address specified in the ADDR\_RING, ADDR\_NODE and ADDR\_MODULE tags. For example assume these tags are currently set to the corresponding values of 1, 4 and 5. The file name that will be loaded is "00100405.C90" or "10405.CFG". OPC90 can load the configuration from one of two file formats. The FILE\_TYPE tag selects the file format to utilize. When FILE\_TYPE is reset (0), OPC90 will load Bailey function block configuration data from files that have a "C90" extension. This is the OPC90 file format. When the file name is included in the command, OPC90 will automatically append the "C90" extension if it is missing. Including the "C90" extension in the file name will override the current FILE\_TYPE tag setting. The "C90" files are loaded from the "ABB" subdirectory unless an alternative directory path is included as part of the file name. When FILE\_TYPE is set (1), OPC90 will load the Bailey function block configuration data from files that have the "CFG" extension. These files are of the format generated by the ABB Bailey CADEWS software and its derivatives. When the file name is included in the command, OPC90 will automatically append the "CFG" extension if it is missing. Including the "CFG" extension in the file name will override the current FILE\_TYPE tag setting. The "CFG" files are loaded from the "ABB" subdirectory unless an alternative directory path is included as part of the file name. Following are some example load commands:

Command	Action
LOAD	Load currently addressed module.
LOAD SLC	Load currently addressed module from a file called "SLC.C90".
LOAD SLC.CFG	Load currently addressed module from a file called "SLC.CFG".
LOAD ROLLER 1 4 6	Load ring 1 node 4 module 6 from a file called "ROLLER.C90".

### Cancel Command

Use this command to cancel a previously started command. Some commands like SAVE or LOAD might take minutes to complete. The cancel command can be sent to cancel the previously started command.

Command	Action
CANCEL	Cancels the previously entered and active command.

### Read Command

Use this command to read an existing function block configured in a Bailey module. Following are some example read commands:

Command	Action
READ	Read currently addressed function block.
READ 30	Read function block 30 from currently addressed module.
READ 40 1 10 20	Read function block 40 from ring 1 node 10 module 20.

### Get Command

Use this command to read default function block data for the any function code supported by a given Bailey module. Note that for some older Bailey modules, the get request must

include a block number value that is consistent with the function code number. For example with a Bailey COM module specifying function code 21 for block 0 will return an error because configuring function code 21 at block 0 is illegal. More current modules will return the default data even if the function code could not be configured at the indicated block number. Following are some example get commands:

Command	Action
GET	Get default specifications for currently addressed function code, block and address.
GET 80	Get default specifications for function code 80 for current address.
GET 30 1 32 30	Get default specifications for function code 30 for ring 1 node 32 module 30.

### Write Command

Use this command to write a new function block to a Bailey module. Remember the module must first be in configure mode before a new block can be written. The steps involved when writing a new function block is to first define it and then write it. First define it by using the READ command to read an existing function block of the same type or using the GET command to read the default settings for the desired function block type. Next change the specification settings by writing to the S01\_VALUE through S63\_VALUE tags. After the specifications have been changed to the desired values enter this command. Following are some example write commands:

Command	Action
WRITE	Write one function block to the currently addressed module.
WRITE 200	Write one function block at block 200 to the currently addressed module.
WRITE 100 10	Starting at block 100 write ten function blocks to the currently addressed module.
WRITE 500 3 6	Starting at block 500 write three function blocks incrementing the written block number by 6 for each write to the currently addressed module. This end result is writing blocks 500, 506 and 512.
WRITE 1000 20 2 1 4 6	Starting at block 1000 write twenty function blocks incrementing the written block number by 2 for each write to ring 1 node 4 module 6.

### Modify Command

Use this command to modify an existing function block in a Bailey module. Remember the module must first be in configure mode before a block can be modified. First use the READ command to read the block to be modified. Next change the specification settings by writing to the S01\_VALUE through S63\_VALUE tags. After the specifications have been changed to the desired new values enter this command. Following are some example modify commands:

Command	Action
MODIFY	Modify one function block in the currently addressed module.
MODIFY 300	Modify one function block at block 300 in the currently addressed module.
MODIFY 50 5	Starting at block 50 modify five function blocks in the currently addressed module.
MODIFY 750 4 3	Starting at block 750 modify four function blocks incrementing the modifying block number by 3 for each modification in the currently

	addressed module. This end result is modifying blocks 750, 753, 756 and 759.
MODIFY 800 10 3 1 10 7	Starting at block 800 modify ten function blocks incrementing the modifying block number by 3 for each modification in ring 1 node 10 module 7.

### Delete Command

Use this command to delete one or more existing function blocks configured in a Bailey module. Remember the module must first be in configure mode before blocks can be deleted. Following are some example delete commands:

Command	Action
DELETE	Delete currently addressed function block.
DELETE 30	Delete function block 30 from currently addressed module.
DELETE 30 40	Delete function blocks 30 through 40 from currently addressed module.
DELETE 100 120 1 10 20	Delete function blocks 100 through 120 from ring 1 node 10 module 20.

### 7.3.9 BLK Configuration Faceplate

The BLK block includes a faceplate that allows the user to configure ABB Bailey controller logic. It incorporates many of the features found in the Bailey TXTEWS and CLS programs. To display the faceplate, right click on the BLK block and select "Faceplate". The following faceplate will be displayed:

**ABB Bailey Controller Configuration**

**Addressing**  
 Ring: 1 Node: 1 Module: 3 Block: 800 FC: 80 Control Station

**Specification Data**

820	S01 Block address of PV >> 820
807	S02 Block address of SP track signal (S29 SP track switch) >> 807
807	S03 Block address of auto signal >> 807
5	S04 Block address of control output track signal (TR) >> 5
687	S05 Block address of control output track switch (TS) >> 687
5	S06 Initial mode of station after start-up >> 5
400	S07 T PV high alarm point in engineering units >> 400
120	S08 T PV low alarm point in engineering units >> 120
75	S09 T PV-SP deviation alarm point in engineering units For console >> 75
800	S10 Signal span of PV in engineering units >> 800
76	S11 Zero value of PV in engineering units >> 76
3	S12 PV engineering units identifier NOTE: For console only >> 3
-5	S13 Signal span of SP in engineering units >> -5
0	S14 Zero value of SP in engineering units >> 0
8	S15 SP engineering units identifier (for DIS console only) >> 8
255	S16 Control station address >> 255

>>

**Command Center**

Read Next Tune Default Write Delete Save Load Exit

Password: [XXXXXXXXXX] File: [ ] .C90 .CFG

Command: None

Request: Block read complete

**Current Mode: Execute**

Change

Execute  
 Configure  
 Initialize  
 Reset

**WARNING**, incorrect use of this faceplate can cause a plant trip or major outage to occur. Make sure you understand ABB Bailey configuration principles and are aware of the current plant operating conditions before using some features of this faceplate.

The faceplate contains three major sections. The boxes with white background indicate possible user input areas, based on a particular configuration activity.

The first section is “Addressing”. This section is used to specify the address of an ABB Bailey module to be configured. The example shows configuration data for a Station (Function Code 80) read from Ring 1, Node 1, Module 3 and Block 800.

The second section is “Specification Data”. This section displays the current specification values and associated descriptions. Tunable specifications are indicated with a “T” following the specification number. Also notice the description includes the current specification value. This is useful when tuning or modifying an existing function block. As a new specification value is typed, the current value is still visible as part of the description. Pressing the <Enter> key will abandon entry of a specification value and reset it back to its original value. Otherwise, it will be updated to the new value after the tune or block write is requested. This section can display up to 16 specifications at a time. For function blocks with more than sixteen specifications, click on the forward and backward (not shown in example) arrow buttons to scroll to the next set of sixteen block specifications.

The last section is “Command Center”. This section is used to request the various configuration activities. Some configuration activities are under password control. The “Password” area is used to enter the password assigned to the BLK block. It displays “????????” to indicate the password has not been entered and “\*\*\*\*\*” when it has. The “Command” area allows expert users to bypass the button commands. See the preceding section entitled “Configuration Command Processing” for details on what commands can be entered. Most users will use the button commands described by the following table.

BUTTON	Description
READ	Enter the address of a block in the addressing section and click on this button to read the function block data.
NEXT	Click on this button to read the next function block data from the currently addressed block.
TUNE	Enter new tunable specification values in the specification data section and click on this button to tune those values within the ABB Bailey controller. Note the module must be “Execute” mode when tuning blocks.
DEFAULT	Enter a function code number in the addressing section and click on this button to retrieve the default specification settings for that function code.
WRITE	Enter the desired specification settings for the function block in the specification data section and click on this button to write the function block data to the ABB Bailey controller. If the block already exists, it will be modified. Note the module must be in “Configure” mode when writing blocks.

DELETE	Enter the address of a block in the addressing section and click on this button to delete it from the ABB Bailey controller. Note the module must be in "Configure" mode to delete blocks.
SAVE	Click on this button to save the currently addressed module. If no file name is entered in the "File" area, the file name will automatically be based on the current ring, node and module address. For the above example it will be "00100103.C90". If the "CFG" radio button is selected it will be "10103.CFG" (to match Bailey CADEWS naming syntax). This convention makes it easy to identify what Bailey module the file corresponds with. Configuration data can be saved in one of two file formats. The format is selected with the "C90" and "CFG" radio buttons. When the "C90" radio button is selected, the function block configuration data is saved into files that have a "C90" extension. This is the OPC90 file format. The "C90" files are stored in the "ABB" subdirectory unless an alternative directory path is included as part of a user inputted file name. When the "CFG" radio button is selected, the function block data is saved into files that have a "CFG" extension. These files must already exist and are of the format generated by the ABB Bailey CADEWS software and its derivatives. The "CFG" files are also stored in the "ABB" subdirectory unless an alternative directory path is included as part of a user inputted file name. Click on the STOP button to stop the file save operation. If the save command is stopped or it fails, the file will not be generated and any original file of that name remains unchanged. Note that if a user inputted file name or directory path is entered that contains spaces, it must be enclosed within quotes ("C:\Program Files\OPC90 Server\ABB\Name With Spaces"). When a user file name is entered it is not necessary to include its extension. The default extension selected by the "C90 / CFG" radio button will automatically be appended to the file. Including the "C90" or "CFG" file extension overrides the default type. Saving a configuration also generates a second file that documents all of the blocks just saved. This list includes the block number, function code type and specification settings. The name of this file is the saved file name with "_BLOCK_LISTING.CSV" appended to it. Microsoft Excel can be used to open the file and look at the blocks contained by the configuration.
LOAD	Click on this button to load the currently addressed module. If no file name is entered in the "File" area, the file name will automatically be based on the current ring, node and module address. For the above example it will be "00100103.C90". If the "CFG" radio button is selected it will be "10103.CFG" (to match Bailey CADEWS naming syntax). This convention makes it easy to identify what Bailey module the file corresponds with. Configuration data can be loaded from one of two file formats. The format is selected with the "C90" and "CFG" radio buttons. When the "C90" radio button is selected, the function block configuration data is loaded from files that have a "C90" extension. This is the OPC90 file format. The "C90" files are found in the "ABB" subdirectory unless an alternative directory path is included as part of a user inputted file name. When the "CFG" radio button is selected, the function block data is loaded from files that have a "CFG" extension. These files must already exist and are of the format generated by the ABB Bailey CADEWS software and its derivatives. The "CFG" files are also found in the "ABB" subdirectory unless an alternative directory path is included as part of a user inputted file name. Click on the STOP button to stop the load operation. Note that if a user inputted file name or directory path is entered that contains spaces, it must be enclosed within quotes ("C:\Program Files\OPC90 Server\ABB\Name With Spaces"). When a user file name is entered it is not necessary to include its extension. The default extension selected by the "C90 / CFG" radio button will automatically be appended to the file. Including the "C90" or "CFG" file extension overrides the default type.
CHANGE	Click on this button to change the operating mode of the currently addressed module to the mode indicated by the radio buttons. The current operating mode of the module is displayed below the mode selection radio buttons.

FILE	Click on this button to open a configuration file open dialog that allows selection of existing .C90 or .CFG files stored in the "C:\Program Files\OPC90 Server\ABB" directory.
------	---



## 7.4 Data Acquisition Analog (DAANG)

The DAANG OPC90 block is used to retrieve the exception reported output from Bailey Data Acquisition Analog blocks (function code 177).

Restrictions: This OPC90 block can be utilized with all Bailey interface types except serial port module (SPM & CPM02).

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey module address.
ADDR_BLOCK	VT_I2	Config/Read	Bailey block address.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey values (0-good, 1-bad).
COMMAND	VT_I2	Read/Write	Command attribute to request Bailey DAANG block as follows (only valid when mode is Auto): 0-Use Auto Input 1-Use Calculated Input 2-Suppress Alarms 3-Un-suppress Alarms 4-Turn Scan Off 5-Turn Scan On
ALARM_CTRL	VT_R4	Read/Write	Alarm control (see FC 177, S20).
SIG_CHANGE	VT_R4	Read/Write	Significant change to force an exception report (see FC 177, S30).
TBA_PERIOD	VT_R4	Read/Write	Time based alarm period (see FC 177, S31).
HI_RATE	VT_R4	Read/Write	High rate of change value (see FC 177, S32).
LO_RATE	VT_R4	Read/Write	Low rate of change value (see FC 177, S33).
TSA_COUNT	VT_R4	Read/Write	Time sequence alarm count (see FC 177, S34).
OUT_HI_LIM	VT_R4	Read	OUT high limit.
OUT_LO_LIM	VT_R4	Read	OUT low limit.
OUT	VT_R4	Read/Write	Current output value received from Bailey.
EU	VT_I2	Read	Engineering Units Code.
EU_TEXT	VT_BSTR	Read	Engineering Units String.
MODE	VT_I2	Read/Write	The mode of the Bailey block. 0 - Manual 1 - Auto
RED_CMD	VT_BSTR	Read/Write	Red tag command (see red tag users section).
RED_TAG	VT_BOOL	Read	Red tag indicator (0 – no, 1 – yes).
RED_USERS	VT_BSTR	Read	Red tag users (current user names or codes).
ALM	VT_BOOL	Read	High or low alarms active indicator.
HI_HI_HI_ACT	VT_BOOL	Read	High high high alarm active indicator.
HI_HI_ACT	VT_BOOL	Read	High high alarm active indicator.
HI_ACT	VT_BOOL	Read	High alarm active indicator.
HI_RATE_ACT	VT_BOOL	Read	High alarm rate active indicator.
LO_RATE_ACT	VT_BOOL	Read	Low alarm rate active indicator.

LO_ACT	VT_BOOL	Read	Low alarm active indicator.
LO_LO_ACT	VT_BOOL	Read	Low low alarm active indicator.
LO_LO_LO_ACT	VT_BOOL	Read	Low low low alarm active indicator.
DV_LIM	VT_R4	Read/Write	Deviation alarm limit.
DV_HI_ACT	VT_BOOL	Read	Deviation high alarm active indicator.
DV_LO_ACT	VT_BOOL	Read	Deviation low alarm active indicator.
HI_HYS	VT_R4	Read/Write	The amount the alarm value must lower below the current high alarm level limit before the associated active current alarm condition clears.
HI_HI_HI_LIM	VT_R4	Read/Write	The setting for the alarm limit used to detect the high high high alarm condition.
HI_HI_LIM	VT_R4	Read/Write	The setting for the alarm limit used to detect the high high alarm condition.
HI_LIM	VT_R4	Read/Write	The setting for the alarm limit used to detect the high alarm condition.
LO_LIM	VT_R4	Read/Write	The setting for the alarm limit used to detect the low alarm condition.
LO_LO_LIM	VT_R4	Read/Write	The setting for the alarm limit used to detect the low low alarm condition.
LO_LO_LO_LIM	VT_R4	Read/Write	The setting for the alarm limit used to detect the low low low alarm condition.
LO_HYS	VT_R4	Read/Write	The amount the alarm value must raise above the current low alarm level limit before the associated active current alarm condition clears.
NEXT_HI	VT_R4	Read	Next high alarm limit to be reached.
NEXT_LO	VT_R4	Read	Next low alarm limit to be reached.
LIMITED	VT_BOOL	Read	Value is limited indicator.
CALC	VT_BOOL	Read	Using calculated input indicator.
OUT_RANGE	VT_BOOL	Read	Out of range detected indicator.
SCAN_OFF	VT_BOOL	Read	Scan disabled indicator.
RATE_ACT	VT_BOOL	Read	Rate alarm indicator.
SUP_ACT	VT_BOOL	Read	Suppress alarm indicator.
VAR_ACT	VT_BOOL	Read	Variable alarm indicator.
HW_FAIL_IND	VT_BOOL	Read	Hardware Failure indicator.
HI_REF	VT_R4	Read	High display reference value (DAANG S1).
MID_REF	VT_R4	Read	Middle display reference value (DAANG S2).
LO_REF	VT_R4	Read	Low display reference value (DAANG S3).
HI_CONST	VT_R4	Read	High constraint limit (DAANG S4).
LO_CONST	VT_R4	Read	Low constraint limit (DAANG S5).

## 7.5 Data Acquisition Digital (DADIG)

The DADIG OPC90 block is used to retrieve the exception reported output from Bailey Data Acquisition Digital blocks (function code 211).

Restrictions: This OPC90 block cannot be used with NETWORK 90 interfaces. It is valid for all INFI 90 interface types.

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey module address.
ADDR_BLOCK	VT_I2	Config/Read	Bailey block address.
OUTLSD0	VT_BSTR	Config/Read	Output state zero logic state descriptor.
OUTLSD1	VT_BSTR	Config/Read	Output state one logic state descriptor.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
FACE_TYPE	VT_I2	Read	Faceplate type (value of DADIG block S17).
ALARM	VT_BOOL	Read	Alarm active indicator.
ALARM_TOGGLE	VT_BOOL	Read	Return alarm toggle indicator.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey values (0-good, 1-bad).
COMMAND	VT_I2	Read/Write	Bailey DADIG block control as follows (note 1): 0-Reset custom input, 1-Set custom input, 2-Select custom input, 3-Select primary input, 4-Select alternate input, 5-Enable alarm suppression, 6-Disable alarm suppression, 7-Clear alarm latch, 8-Disable exception reports from this block, 9-Enable exception reports from this block, 10-Force an exception report from this block.
IN_CTRL	VT_R4	Read/Write	Input conditioning mode control (see FC 211, S6).
IN_TREF	VT_R4	Read/Write	Input conditioning time reference, seconds (see FC 211, S7).
IN_CREF	VT_R4	Read/Write	Input conditioning digital filter count reference (see FC 211, S8).
ALARM_CTRL	VT_R4	Read/Write	Alarm mode control (see FC 211, S9).
ALARM_TREF	VT_R4	Read/Write	Alarm time reference, seconds (see FC 211, S10).
ALARM_TCOUNT	VT_R4	Read/Write	Alarm digital filter transition count reference (see FC 211, S11).
RTP_REF	VT_R4	Read/Write	Return alarm/de-alarm time period reference (see FC 211, S12).
OUT	VT_BOOL	Read	Bailey discrete output value.
OUT_TEXT	VT_BSTR	Read	Bailey discrete output value as a string.

OVR_STATUS	VT_BOOL	Read	Status override indicator.
SUSPECT	VT_BOOL	Read	Alarm suspect indicator.
E_STOP	VT_BOOL	Read	Emergency stop indicator.
IN_SELECT	VT_BOOL	Read	Input is selectable indicator (note 2).
IN_ALTERNATE	VT_BOOL	Read/Write	Alternate input active indicator / request (note 2).
IN_CUSTOM	VT_BOOL	Read/Write	Custom input active indicator / request (note 2).
IN_PRIMARY	VT_BOOL	Read/Write	Primary input active indicator / request (note 2).
LATCH	VT_BOOL	Read/Write	Alarm latch indicator / reset request (note 3).
SUPPRESS	VT_BOOL	Read/Write	Alarm suppress indicator / set / reset (note 4).
XRP_OFF	VT_BOOL	Read/Write	Exception report off indicator / set / reset / request (note 5).

Notes:

- 1.) The COMMAND attribute can be used to send various commands to the Bailey DADIG block. Specifically, it allows selection of the CUSTOM input state, what input should currently be used, control of alarm suppression, resetting the alarm latch and control of exception reporting. Note that writing to the attributes that report the current state of that block feature can also control these features (notes 2 – 5).
- 2.) When the IN\_SELECT attribute is TRUE (1), the DADIG block can be commanded to use its various inputs. The IN\_ALTERNATE, IN\_CUSTOM and IN\_PRIMARY attributes indicate the current input being used by the block. They can also be used to command the block to use a different input. For example assume the current input is primary, then IN\_PRIMARY will be one, IN\_ALTERNATE and IN\_CUSTOM will be zero. When a one is written to IN\_ALTERNATE, the DADIG block will begin using the alternate input. Note that writes will be rejected if the IN\_SELECT attribute is not TRUE (1).
- 3.) The DADIG block alarm latch is indicated by this attribute. By writing FALSE (0) to this attribute, the latch will be cleared.
- 4.) This attribute indicates alarm suppression is in effect. It can also be used to set or reset alarm suppression (only if enabled by the DADIG block configuration) by writing a respective TRUE (1) or FALSE (0) to this attribute.
- 5.) This attribute indicates whether or not exception reporting for this block has been disabled. If XRP\_OFF is TRUE (1), exception reporting is disabled. This attribute can be used to turn exception reporting off or on. Write TRUE (1) to it to turn exception reporting off. Write FALSE (0) to it to enable exception reporting. When exception reporting is off and TRUE (1) is written to this attribute, an exception report force command will be sent. This message causes the block to exception report once but still leave exception reporting off.

## 7.6 Data Block (DATA)

The purpose of the OPC90 DATA block is to enable OPC clients to communicate information with each other through values they read / write to the various tags supported by this block. The DATA block supports VT\_BOOL, VT\_I1, VT\_I2, VT\_I4, VT\_UI1, VT\_UI2, VT\_UI4, VT\_R4, VT\_R8 and VT\_BSTR tag values. It also includes a general purpose high resolution timer function and Boolean swap state timer function. The high resolution timer units can be setup for seconds, minutes, hours or days. It can also be setup as an up or down timer that automatically restarts or must be manually restarted. The Boolean swap state timer flops its value between true and false at the specified period (seconds). All of these timer functions are controllable from OPC client accessible OPC tags. Whenever the database is saved, the current OPC client timer settings are included in the saved data and will be restored when OPC90 is restarted.

Restrictions: This OPC90 block counts against the total block license but does not consume any space or bandwidth within the Bailey interface.

TAGNAME	TYPE	ACCESS	DESCRIPTION
BOOL	VT_BOOL	Read/Write	Boolean, VT_BOOL (0 = false, not zero = true).
BSTR	VT_BSTR	Read/Write	String, VT_BSTR.
I1	VT_I1	Read/Write	Signed I1, VT_I1 (-128 to 127).
I2	VT_I2	Read/Write	Signed I2, VT_I2 (-32,768 to 32,767).
I4	VT_I4	Read/Write	Signed I4, VT_I4 (-2,147,483,648 to 2,147,483,647).
UI1	VT_UI1	Read/Write	Unsigned I1, VT_UI1 (0 to 255).
UI2	VT_UI2	Read/Write	Unsigned I2, VT_UI2 (0 to 65,535).
UI4	VT_UI4	Read/Write	Unsigned I4, VT_UI4 (0 to 4,294,967,295).
R4	VT_R4	Read/Write	Real 4, VT_R4.
R8	VT_R8	Read/Write	Real 8, VT_R8.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
TIMER	VT_R8	Read/Write	Running timer (in units setup by TIMER_CTRL).
TIMER_CTRL	VT_UI1	Read/Write	Timer control (+0 secs, +1 mins, +2 hours, +3 days, +4 auto reset, +8 down timer).
TIMER_ALARM	VT_BOOL	Read	Timer alarm indicator (true = timed out).
TIMER_LIMIT	VT_R8	Read/Write	Timer limit.
SWAP	VT_BOOL	Read/Write	Swaps state between true / false.
SWAP_TIME	VT_R8	Read/Write	Time between each state swap (seconds).

## 7.7 Device Driver (DD)

The DD OPC90 block is used to retrieve and control the exception reported output from Bailey Device Driver blocks (function code 123).

Restrictions: This OPC90 block can be utilized with all Bailey interface types except serial port module (SPM & CPM02).

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey module address.
ADDR_BLOCK	VT_I2	Config/Read	Bailey block address.
F1LSD0	VT_BSTR	Config/Read	Feedback 1 state zero logic state descriptor.
F1LSD1	VT_BSTR	Config/Read	Feedback 1 state one logic state descriptor.
F2LSD0	VT_BSTR	Config/Read	Feedback 2 state zero logic state descriptor.
F2LSD1	VT_BSTR	Config/Read	Feedback 2 state one logic state descriptor.
OUTLSD0	VT_BSTR	Config/Read	Output state zero logic state descriptor.
OUTLSD1	VT_BSTR	Config/Read	Output state one logic state descriptor.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
FACE_TYPE	VT_I2	Read	Faceplate type (value of DD block S10).
DISC_ACT	VT_BOOL	Read	Alarm active indicator.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey values (0-good, 1-bad).
OUT	VT_BOOL	Read/Write	Bailey discrete output value.
OUT_TEXT	VT_BSTR	Read/Write	Bailey discrete output value as a string (see note 1).
MODE	VT_I2	Read/Write	The mode of the Bailey block. 0 - Manual 1 - Auto
MODE_LOCK	VT_BOOL	Read	Indicates mode is locked in auto (0-no, 1-yes).
RED_CMD	VT_BSTR	Read/Write	Red tag command (see red tag users section).
RED_TAG	VT_BOOL	Read	Red tag indicator (0 – no, 1 – yes).
RED_USERS	VT_BSTR	Read	Red tag users (current user names or codes).
F1	VT_BOOL	Read	State of feedback number 1.
F1_TEXT	VT_BSTR	Read	State of feedback number 1 as a string.
F2	VT_BOOL	Read	State of feedback number 2.
F2_TEXT	VT_BSTR	Read	State of feedback number 2 as a string.
BAD_FEEDBACK	VT_BOOL	Read	Feedback bad indicator.
OVR_STATUS	VT_BOOL	Read	Override status indicator.

### Notes:

- 1.) Text strings can be written to this tag to control the DD output. Valid text strings are the OUTLSD0 and OUTLSD1 settings along with “0”, “1”, “Off”, “On”, “Reset”, “Set”, “Zero”, “One”, “False” and “True”. Writing any of these strings is case insensitive.

## 7.8 Device Definition Block (DEVICE)

The DEVICE Definition OPC90 block is used to declare an instance of the Bailey driver and define its interfacing data. This block must be defined for each Bailey interface or redundant pair of interfaces that the OPC90 Server is to communicate with. The DEVICE block has a number of tags used to monitor driver communication health and statistics. Italicized names indicate attributes configured within the OPC90 Server, which cannot be connected to via OPC. These attributes are the device properties. Device properties are configured when the block is first added. They are also be changed by “right clicking” on the device block and selecting properties.

<b>ATTRIBUTE/ TAGNAME</b>	<b>TYPE</b>	<b>ACCESS</b>	<b>DESCRIPTION</b>
<i>SCHEME</i>	8 Bit UINT	Config	Desired communication scheme (see note 1): Single interface, Single interface redundant channels, Dual interfaces.
<i>Primary Port</i>	String	Config	Name of the COM, SCSI (see SCSI communication sub-section) or TCP (see TCP communication sub-section) port attached to the primary interface channel.
<i>Secondary Port</i>	String	Config	Name of the COM, SCSI (see SCSI communication sub-section) or TCP (see TCP communication sub-section) port attached to the secondary interface channel. Note that this attribute is ignored if the SCHEME attribute is set to single interface.
<i>MAX OUTPUTS</i>	16 Bit UINT	Config	Maximum number of output blocks (AOLs, DOLs, ODDs, OMSDD, ORCMs, ORMCs, ORMSCs and OSTNs) the driver should reserve point indices (block numbers) for within the Bailey interface device. It is important to note that these indices are specified as part of the AOLs, DOLs, ODDs, OMSDD, ORCMs, ORMCs, ORMSCs and OSTNs block definition and must always fall in the range of one to the maximum outputs defined by this attribute.
<i>WATCHDOG</i>	8 Bit UINT	Config	Desired watchdog timer to be initiated between the driver and Bailey interface device. The timer is expressed in 2.5 second counts. A value of zero disables the watchdog timer. The maximum value is 255 which is equivalent to 637.5 seconds. When the watchdog timer is enabled, the Bailey interface device will remove itself from the communication loop if the elapsed time in which the driver communicates with it exceeds the watchdog timer value. Driver output block values (AOLs, DOLs, ODDs, OMSDD, ORCMs, ORMCs, ORMSCs and OSTNs) being received by Bailey controllers will thereafter automatically be marked as bad quality by the Bailey system.
<i>INPUT XRP UPDATE</i>	32 Bit UINT	Config	Frequency in milliseconds at which exception report availability is read from the Bailey interface

			device. Settings in the range of 500 to 3000 milliseconds are common.
<i>OUTPUT XRP UPDATE</i>	32 Bit UINT	Config	Frequency in milliseconds at which output exception reports are written to the Bailey interface device. Output exception reports are values written to AOL, DOL, ODD, OMSDD, ORCM, ORMC, ORMSC and OSTN blocks by OPC clients. Settings in the range of 500 to 3000 milliseconds are common.
<i>MAX KEYS PER PCU</i>	32 Bit UINT	Config	Messages that read block outputs and block specifications use a CIU mechanism called keyed messages. These messages allow the CIU to be working on getting that data while still responding to other requests such as exception report reading. This setting determines the maximum number of keys that will be used per PCU for keyed type messages. The settable range is 0 or 4-32. When set to 0, all possible keys relevant for the particular type of CIU device could be used. For most applications, the default setting of 4 is optimal.
<i>BLOCK SPECIFICATION READ PERIOD (SECONDS)</i>	32 Bit UINT	Config	Defines the period in seconds for how often read block specification messages are generated. This period along with the MAX BLOCK SPECIFICATIONS READ PER PERIOD parameter determine how often read block specification messages are sent for each block. So for example, setting the period to 15 seconds and MAX BLOCK SPECIFICATIONS READ PER PERIOD to 1 will result in a request for a block specifications to be sent every 15 seconds. If there are 100 blocks in the configuration requiring block specifications to be read it will take 1500 seconds to cycle through all of the blocks.
<i>MAX BLOCK SPECIFICATIONS READ PER PERIOD</i>	32 Bit UINT	Config	Defines the maximum block specification read rate per period for STN and SPEC blocks (DAANG, DADIG, DD and MSDD are read once on startup). The settable range for this parameter is 1-32. The period is set using the BLOCK SPECIFICATION READ PERIOD parameter. Not all CIUs and ABB Bailey systems are capable of reading at 32 per second. Most systems should not attempt to read block specifications at such a high rate. Setting it too high could cause nodes to become busy (Infi 90 systems) or go offline (Network 90 systems). Also higher settings may cause a slowdown in a PCUs ability to report exception reports. Watch the MSG_QUEUE tag and if it continues in the double digits for prolong periods, lower this setting. Unless faster block spec read cycles are absolutely required, it is best to leave this setting at the default value or lower it even more. So for example, setting this parameter to 2 and the BLOCK SPECIFICATION READ PERIOD to 60 seconds will result in two requests for a block specifications to be sent every 60 seconds. Let's say there are only 10 STN blocks in the



			configuration. It will take 300 seconds to cycle through reading the PID gains associated with those STN blocks.
<i>MAX GMI PER SECOND PER PCU NODE</i>	32 Bit UINT	Config	GMI stands for general message interface. They are messages such as block output polling, block configuration specification reading, problem report reading and others. Basically any message not related to exception report collection. Sending too many GMIs to a PCU node could cause that node to slow down the number of exception reports it needs to send. This setting can be used to throttle the number of GMI messages that OPC90 is allowed to send per second to PCU nodes. Normal use of OPC90 does not need this parameter to be changed. When used with MUXCIU blocks servicing DBDOC and Composer it may need lowered if exception report slowdowns are noticed while Composer is doing downloads or verifies. This occurrence in most cases won't be seen when using serial CIU communication but could happen for SCSI CIUs.
<i>ENABLE TURBO POLLING</i>	Boolean	Config	Turbo polling significantly increases the amount of data that can be polled using the POLL block, the STN.KFAST feature and poll requests received via the MUXCIU block. Turbo polling lowers the loading effect of block output polling by consolidating block output reads and thus reducing the total number of messages required to poll the data. This feature works for both SCSI and serial interfaces but is most effective with SCSI even for systems having high exception report traffic. Its effectiveness for serial interfaces with high exception report traffic is much more pronounced when using the redundant channel communication scheme. Turbo polling does not support getting alarm bits from blocks that have such data. Therefore it should only be disabled if the alarm bit polling information is required.
<i>NON-KEYED MESSAGING</i>	Boolean	Config	This setting causes all GMI type messaging to occur using non-keyed messages. Use of this option should be limited to SCSI CIU configurations containing MUXCIU blocks connected with G.Michaels DBDOC and / or ABB Composer. Its use with databases receiving exception reported values works best for the redundant channel or dual interface communication scheme. Use of this option with plant loop system (NETWORK 90) is not recommended.
<i>LOCK</i>	Boolean	Config	Supervisory control write lock. When True (1), locks out all operator write requests (supervisory control write) from the other OPC90 Server blocks that support write capability. If write locks are only needed on an individual block basis don't use this setting but instead using the "read only" setting when configuring each individual block.

<i>ESTABLISH ONLINE</i>	Boolean	Config	Commands interface on-line before establishing the points. Use this option for heavily loaded Network 90 systems. It only needs to be enabled when PCUs are observed to toggle between the online and offline state when OPC90 Server is starting.
<i>SCREENING</i>	Boolean	Config	Utilize the interface exception report screening option. This option helps increase throughput by instructing the interface to not report unchanged exception reports received from the sender as a result of its maximum exception report time limit expiring.
<i>ENHANCED ANALOG PRECISION</i>	Boolean	Config	Enables analog data exchange with exception reported blocks using REAL4 representation instead of REAL3. REAL4 representation increases the accuracy of analog values to 7 digits. REAL3 values have an accuracy of 5 digits. Enabling this option is only valid for INFI90 systems. It is ignored when the detected system type is NETWORK 90 or Command Series. This option adds an extra byte to each analog value exchanged with the CIU. The result is a very slight reduction in the maximum number of exception reports that can be exchanged per second. Most users should enable this option. It has been made optional to maintain compatibility with existing OPC90 installations.
<i>MSDD PULSE OUT HANDLING</i>	Boolean	Config	Enables additional logic that handles MSDD blocks configured for pulsed output operation. The MSDD pulse output time specification (S22) is read on startup. If non-zero, whenever changes to the requested state are made, the OPC90 MSDD block REQ_STATE tag will be reset to zero after the configured pulse output time has transpired. Remember, any changes made to MSDD block pulse output times after startup of OPC90 are not recognized until OPC90 is restarted.
<i>TIME SYNCING</i>	Boolean	Config	Supports defining the type of ABB Bailey system and time syncing related options (see note 2).
<i>ACCURACY RATING</i>	BYTE	Config	Accuracy rating assigned to the PC running OPC90 and its CIU. This rating can be 0 – 255. CIU nodes with the highest accuracy rating are allowed to become the time sync master if time syncing is enabled for that node. This setting can be used to define which OPC90 should be favored as the time sync master node (see note 2d).
<i>STN FAST Update</i>	32 Bit UINT	Config	Defines the fast update interval in minutes and poll rate in milliseconds for all STN blocks belonging to this device. When a STN.K* attribute is written or STN.KFAST (see STN block description) is set the server automatically polls for STN.PV and STN.CO values for the interval and rate defined by this setting. This feature is very useful for loop tuning software. Setting the fast update interval to zero will disable it.

<i>Set .KFAST on K* tag writes</i>	Boolean	Config	When set, any write to the PID tuning tags (.K*, OUT_HI_LIM AND OUT_LO_LIM) will automatically turn .KFAST on.
<i>Represent STN Mode As 0 - 5</i>	Boolean	Config	Normally the STN block represents mode in the range of 0 – 2 where 0 = manual, 1 = auto and 2 = cascade / ratio. This representation is the same regardless of the current station operating level (local or computer). The station level is monitored and controlled using the STN_LEVEL tag. Setting this option changes the STN mode representation to be in the range 0 – 5 where 0 = local manual, 1 = local auto, 2 = local cascade / ratio, 3 = computer manual, 4 = computer auto and 5 = computer cascade / ratio. This option allows OPC clients to control STN mode and level via the STN.MODE tag.
<i>AUTOMATIC SEND STN CPU_OK</i>	Boolean	Config	Instructs OPC90 to send CPU_OK messages to any STN block that is in computer level control. The frequency of these messages will be ¾ of the current STN block computer watchdog time setting.
<i>NODE01 to NODE63</i>	Array of 8 Bit UINTs	Config	Defines the node map required for ABB Bailey Network 90 systems when the time synchronization option has been enabled (see note 2). For each possible node number, one of the following choices must be selected; Vacant, 1.4K Tag OIU, PCU, CIU01, Other. The node map must be accurately defined for time synchronization to function correctly on Network 90 based systems. These attributes are ignored when time synchronization is disabled or an Infi 90 system has been specified or detected.
<i>SIMULATION</i>	Boolean	Config	Two types of simulation can be selected. Selecting “Device and input blocks”, simulates communication with the interface and reception of values. The type of value simulation is configurable for each block attribute by right clicking on the attribute and selecting properties and then the simulation signal type. The selections are sine, ramp, random and constant. Selecting the “AOL and DOL block input” simulation option causes communication with the interface to occur normally with the input values to the AOL and DOL blocks to be automatically updated with simulated values that would normally be provided by an OPC client. This simulation option also allows configuration of the percent of the total values to change per second.
<i>TAG STARTUP</i>	Array	Config	Picks the initial OPC status that will be reported to OPC clients for all of the tag values until the Bailey interface has been initialized with the configured database and has begun to report the actual values.
<i>SET LAST TAG VALUE</i>	Boolean	Config	On startup the tag values associated with AIL, AOL, DIL and DOL blocks will be initialized to the last saved value before OPC90 was last stopped.

			For this option to be effective the Auto Save database feature must also be enabled.
<i>AUTO DISCONNECT</i>	Boolean	Config	With this option set, all block values that are not connected to an OPC client or currently being viewed in the program window or faceplate will be automatically disconnected. If the exception report for the block value goes into or out of alarm condition it will automatically be connected by the CIU, reported and then disconnected by OPC90. This option is very useful for large databases that have blocks that OPC clients aren't interested in getting data from. It saves communication bandwidth but keeps the blocks available as needed in the future. Some redundancy schemes that may rely on both servers having the same data should not run with this option enabled. This option also applies to polled type of block values such as SPEC and POLL blocks. It has no effect on block output values being polled via MUXCIU block connections.
<i>STARTUP Complete</i>	Boolean	Config	Normally the STARTUP device tag resets when all of the points are established in the CIU. Setting this option delays the STARTUP reset condition until after all of the blocks receive initial values from the ABB Bailey DCS.
<i>Set Bad Output Quality on Max Exception Timeout</i>	Boolean	Config	The QUALITY tag will indicate bad for AOL, DOL, ODD, OMSDD, ORCM, ORMC, ORMSC and OSTN blocks when writes to the input tags do not occur within the block's exception report output max time setting. When this occurs the bad quality condition is also written to the CIU and therefore propagated to the users of these data points in the ABB Bailey system. This property should only be used when OPC clients writing the input values to these blocks does so on a regular basis even when the value has not changed (see next property).
<i>Any Input Write Applies to All Output Blocks</i>	Boolean	Config	With this option set, an input write to any AOL, DOL, ODD, OMSDD, ORCM, ORMC, PRMSC and OSTN block reset the internal bad quality maximum timer for all of these blocks. This option is useful when digitals are not changing and therefore not being written on a regular basis whereas analogs usually are. A write to any input value becomes the indicator the OPC client is still alive.
<i>Allow turbo reads for bad quality blocks</i>	Boolean	Config	Turbo polling messages return values of zero when the quality of the block being read is bad. To enable the ability to see polled bad quality values, the turbo polling logic drops back to polling them with single block output reads. If it is more desirable to not see these values and maintain the efficiency that reading them with turbo polling provides, this operation can be overridden by enabling this property.

<i>CONTINUOUS OPC UPDATES</i>	Boolean	Config	Instructs OPC90 to report all data change tags at the data change cycle even when they have not changed from the last reported value. This option is useful when interfacing to OPC clients that mark OPC data stale or bad when it is not received on a regular basis. Most OPC clients do not need this option to be set.
<i>XRP WRITE CONFIRM</i>	Boolean	Config	Supervisory control writes to DD, MSDD, RCM, RMC, RMSC and STN blocks are sent directly to the Bailey system without updating the OPC tag database. The OPC tag database gets updated when the Bailey exception report is received confirming the write was received and accepted. This option may be useful when implementing OPC client supervisory control ramping functions. Also some Bailey block logic could have supervisory control locks in place. These locks prevent the block from completing the request but no exception report is generated indicating the request was rejected.
<i>GROUP WRITES</i>	Boolean	Config	When set, output exception reports generated from AOL, DOL, ODD, OMSDD, ORCM, ORMC, ORMSC and OSTN blocks will be written using the output group commands. Up to fifty values per command can be written. OPC90 automatically determines the group size based on current loads and output block updates. This setting optimizes output exception reporting and should be used when large numbers of output blocks are configured. When this option is not set, each value is sent with individual commands. The rate at which outputting occurs is also configurable. See the OUT XRP UPDATE setting.
<i>DEBUG LOG</i>	Boolean Array	Config	Selected by right clicking on the DEVICE block. Enables various logging features that may be useful for diagnosing driver operational performance. When any of the logging features are enabled, daily log files are generated in the OPC90 program directory. Leaving all but the error option enabled for extended periods of time can consume a lot of disk space. The four logging options supported are: Errors – post miscellaneous errors Events – post communication events Sends – post messages sent to interface. Receives – post messages received from interface.
<i>LOG_DAYS</i>	VT_I4	Config	Number of days the debug logs should be retained. Logs older than the number of days indicated by this setting will be automatically deleted.
<i>MY_BLOCK_TYPE</i>	VT_BSTR	Read	Indicates the OPC90 block type.
<i>MESSAGE</i>	VT_BSTR	Read	Text message indicating overall status of the device.

POINT_TOTAL	VT_I4	Read	Total blocks this DEVICE block has been requested to exchange data with the Bailey system.
ALL_STATUS	VT_I4	Read	Primary and secondary communication channel status (AND value of the two status) where a value of zero indicates at least one is good and one means both are bad.
PRI_STATUS	VT_I4	Read	Primary communication channel status. A value of zero indicates good and one means bad.
SEC_STATUS	VT_I4	Read	Secondary communication channel status. A value of zero indicates good and one means bad or not used.
ONLINE	VT_I4	Read	CIU online state (0-No, 1-Yes).
DATA_READY	VT_I4	Read	Indicates data should now be available (0-Yes, 1-No). This tag is a combination of the logical or of tags .ALL_STATUS, .STARTUP, along with not of .ONLINE. Its purpose is for simplistic OPC clients that only allow one tag to be referenced as the decision maker for its failover mechanism.
BLOCK_CYCLE	VT_I4	Read	This is a counter that bumps each time all blocks requiring background specification reads has been completed. The time between each counter update can be shorten or lengthen with adjustment to the device "Maximum Block Specifications Read / Sec" setting. On startup, the initial cycle takes much longer because all DAANG, DADIG, DD, MSDD, SPEC and STN blocks must have their block specifications read. Thereafter, only the STN and SPEC blocks require reading on an ongoing basis.
MSG_QUEUE	VT_I4	Read	This value indicates how many messages have been queued and waiting for available keys to get sent. It should hover in a single digit number. The indication that the "Maximum Block Specifications Read / Second" is set to a value higher than the particular CIU type can handle will be the queue value climbing into double digits. Other polling requirements such as POLL, SPEC and STN blocks or BLK blocks doing a save or load may also be responsible for more temporarily queued messages which is normal and expected.
MSG_TOTAL	VT_I4	Read	Running count of total messages being exchanged with the Bailey interface.
MSG_RATE	VT_I4	Read	Messages per second being exchanged with the Bailey interface.
XRPS_TOTAL	VT_I4	Read	Running count of total exception report specifications received from the Bailey interface.
XRP_TOTAL	VT_I4	Read	Running count of total exception reports exchanged with the Bailey interface.
XRP_RATE	VT_I4	Read	Exception reports per second being exchanged with the Bailey interface.
POLL_TOTAL	VT_I4	Read	Running count of total values polled from the Bailey interface.
POLL_RATE	VT_I4	Read	Polled values per second being read from the Bailey interface.
POLL_OUT_TOTAL	VT_I4	Read	Total read block output values read

POLL_OUT_RATE	VT_I4	Read	Read block output values / second
NAK_TOTAL	VT_I4	Read	Running count of total negative acknowledgments received from the Bailey Interface.
MY_DEVICE	VT_BSTR	Read	Text message indicating the ABB Bailey interface type (i.e. INICI03, CIC01, etc) and firmware revision.
MY_RING	VT_I4	Read	Ring address of the Bailey interface.
MY_NODE	VT_I4	Read	Node address of the Bailey interface.
SHADOW_OK	VT_BOOL	Read	Shadowing OK (0-Good, 1-Bad or Disabled)
SHADOW_MASTER	VT_BOOL	Read	Currently Shadow Master (0-No, 1-Yes)
SHADOW_PRI_STATUS	VT_I4	Read	Shadowed CIU primary communication channel status. A value of zero indicates good and one means bad or not used.
SHADOW_SEC_STATUS	VT_I4	Read	Shadowed CIU secondary communication channel status. A value of zero indicates good and one means bad.
SHADOW_MY_RING	VT_I4	Read	Shadowed CIU ring address of the Bailey interface.
SHADOW_MY_NODE	VT_I4	Read	Shadowed CIU node address of the Bailey interface.
SHADOW_MY_DEVICE	VT_BSTR	Read	Text message indicating the shadowed ABB Bailey interface type (i.e. INICI03, CIC01, etc) and firmware revision.
SHADOW_ONLINE	VT_I4	Read	Shadow CIU online state (0-No, 1-Yes).
SYNC_DCS	VT_BOOL	Read/Write	Set this tag to request the DCS to be time synced to the current PC clock time (see note 2).
SYNC_RING	VT_I4	Read	Ring address of current time sync master (see note 2).
SYNC_NODE	VT_I4	Read	Node address of current time sync master (see note 2).
IMPORT_FILE	VT_BSTR	Read/Write	CSV file to import (see note 3)
IMPORT_READ	VT_BOOL	Read/Write	CSV file import control flag (see note 3)
IMPORT_RESULT	VT_BSTR	Read	Result of CSV import request (see note 3)
TURBO_ACTIVE	VT_I4	Read	Total active blocks in the turbo poll database
TURBO_TOTAL	VT_I4	Read	Total blocks in the turbo poll database
TURBO_STALE	VT_I4	Read	Total active turbo polled blocks that are stale

#### Notes:

- 1.) The DEVICE Definition block is designed to support communication on one or two RS232 ports to a single Bailey interface or two RS232 or SCSI ports to redundant Bailey interfaces. The first redundant scheme is single interface redundant channel. This scheme provides two RS232 communication paths to a single Bailey interface. All Bailey interfaces except (SPM, CPM, CIC, CIU01, INPCI01 and INICI03) have two RS232 ports available for communication. The second port is switch selectable between a "utility" port and computer communication port. Setting it to a computer communication port allows it to be used for the single interface redundant channel communication scheme. With single interface redundant channel, the driver issues exception report read requests to both channels resulting in increased throughput. If loop tuning software has one or more STN blocks in fast update mode, exception report reading on the second channel is temporarily stopped and instead it is used for polling the STN block process variable and control outputs. This feature helps facilitate tuning of fast loops that typically require 4 or more updates per second even when the active database is very large. When the STN blocks are taken out of fast update mode, exception report collection on the second channel will resume. The second redundant scheme is dual interfaces single channel. This scheme utilizes two Bailey interfaces. Each Bailey interface can be assigned the same node address on the Bailey communication highway or given unique addresses. When the Bailey interfaces are configured for the same node address, the driver commands the primary interface online to the Bailey

communication loop and commands the secondary interface offline. The database is downloaded to both interfaces so the secondary can be considered to be in a “warm” standby mode, ready to be commanded online if the primary interface fails. When the Bailey interfaces are configured for different node addresses the driver commands both interfaces online receiving exception report data from each. This increases the effective data throughput. The decision of Bailey node address assignment should be based on whether or not any OPC90 Server AOLs, DOLs, ODDs, OMSDD, ORCMs, ORMCs, ORMSCs or OSTNs blocks are to be utilized. When they are used, the Bailey interfaces should be assigned the same node address. Otherwise, when AOLs, DOLs, ODDs, OMSDD, ORCMs, ORMCs, ORMSCs and OSTNs are not used, unique node addresses should be assigned to take advantage of the extra throughput. When the interfaces are assigned unique addresses, exception report collection will be temporarily stopped on the secondary interface when loop tuning software has one or more STN blocks in fast update mode. This allows the use of that interface for the required STN fast update poll commands. This feature helps facilitate tuning of fast loops that typically require 4 or more updates per second even when the active database is very large. When the STN blocks are taken out of fast update mode, exception report collection on the second interface will resume.

- 2.) The DEVICE definition block support selection of several time syncing options. Note that the ability for time syncing to function correctly depends on several factors. The most important being that the ABB Bailey interface supports the time syncing feature. Those interfaces that do not support time syncing are NSPM01, IMSPM01, IMCPM02, NCIU01 and INPCI01. Note also that time syncing will not work properly on Network 90 systems that have 1400 tag OIUs present on the ABB Bailey Plant Loop. The SYNC\_DCS, SYNC\_RING and SYNC\_NODE tags are only updated when one of the following time sync options are enabled. When they are, setting the SYNC\_DCS tag can be used to force the DCS to be time synced to the PC clock time. This also automatically occurs when the PC clock is changed by more than 2 minutes.
  - a.) To enable time syncing the ABB Bailey system type must first be specified. Choices are Infi 90 and Network 90. When the system type is Network 90 the system node map must also be configured. This node map defines the addresses and type of nodes present on the ABB Bailey Plant Loop. Failure to completing and accurately configure the node map could result in malfunction of the time syncing feature. All other system types beside Network 90 do not require definition of the node map. Note the Infi 90 system type should be specified when the interface is a NCIC01.
  - b.) When the ABB Bailey DCS is an Infi 90 system the DEVICE block allows definition of a property called “Accuracy rating”. This property is used to define the clock accuracy rating of its PC / CIU. Accuracy rating is used when time syncing is enabled. OPC90 will allow the node with the highest accuracy rating to become the time sync master. It will not accept time from any node with an accuracy rating less than its own. The accuracy rating can be set to any value between 0 – 255. Systems with OIS consoles interpret accuracy rating as 0 = lowest accuracy, 3 = low accuracy battery backed clock, 6 = high accuracy battery backed clock and 12 = satellite clock. For these systems, setting the accuracy rating higher than 12 will guarantee, OPC90 becomes the time sync master node.
  - c.) Two time syncing options are available. The first is “PC Set DCS time” This option enables OPC90 to set the ABB Bailey DCS time to match the PC time. The second is “PC Get DCS time”. This option enables OPC90 to set the PC time to match the ABB Bailey DCS time. If neither of these options are enabled OPC90 does not participate in time synchronization. Here are the details concerning functionality of these options.
    - i. PC Set DCS time
      - Becomes time sync master if current accuracy rating is higher than current time sync master.
      - When requested, becomes time sync master if its accuracy rating is equal to current time sync master.
      - Defers to other nodes to be time sync master when their rating is higher.



- Never sets DCS time from PC time when option not enabled.
- ii. PC Get DCS time
  - Sets PC time from DCS time when its accuracy rating is lower or equal to current time sync master.
  - Will not set PC time from DCS time when its accuracy rating is greater than accuracy rating of current time sync master.
  - Never sets PC time from DCS time when option not enabled.
- d.) When OPC90 starts up with these options set it determines if another ABB Bailey node is currently the time sync master and its accuracy rating. If another master exists with equal or greater accuracy rating, OPC90 does not attempt to become the time sync master. When it determines it has a better accuracy rating OPC90 becomes the master by setting the DCS time equal to the PC time. OPC90 continues to monitor the SYNC\_DCS tag for being set and PC clock for changes in time that exceed 2 minutes. If either occurs, and OPC90 has the highest or equivalent accuracy rating it will become the time sync master by setting the DCS time to the PC time. If another node becomes the time sync master OPC90 will set the PC time from the DCS time when the “PC Get DCS time” option is set. This option is handy for keeping the PC time equal to the DCS time maintained by another time sync source. This option can be disabled when the PC clock is being updated by a domain controller or other source not associated with the DCS time.
- e.) When the ABB Bailey DCS is an Infi 90 system and either time syncing option is selected, the “Use DCS Timestamp” option may also be enabled. This option causes the OPC time stamps associated with each data value to be set to the time stamp generated within the DCS. Normally per the OPC foundation specification, OPC90 generates the OPC value time stamp when the value is sent to the OPC client. This option allows more precise time stamping data to be associated with each value. It should be noted this occurs at the expense of slightly reduced exception report throughput since each exception reported value received from the interface will also include a 6 byte DCS time stamp. This option provides sub second time stamp accuracy versus the 1-2 second time stamp accuracy when not enabled. The “Allow DCS timestamp Override” option can be enabled to comply with OPC foundation specification concerning sync reads. When enabled, OPC90 overrides the DCS timestamps to the current PC timestamp for all points read by OPC clients using the sync read function.
- f.) The “Y2k Time Warping” option allows the PC time to remain in the present (year 2000 through 2021) while time syncing the ABB Bailey DCS to calendar years prior to year 2000 in the range of 1980 to 1999. It is rare to use this option since the majority of systems have been updated to not have any Y2K issues. OPC90 will set the DCS time to pre 2000 years that have calendars that match the present day calendar. This option is useful for ABB Bailey systems having controller firmware that is not Y2k compatible but having control events that must occur based on a specific day of the week (Monday, Tuesday, Wednesday, etc). Note that this option is mutually exclusive with the “use DCS Timestamp” option previously discussed. There are two important factors that must be considered when using this option. The first is that it must be enabled for all PCs running OPC90 that have the time syncing option enabled. The second is its use is not recommended for systems that also have other nodes present (like ABB Bailey consoles) that are configured to participate in time syncing or are using the ABB Bailey distributed trending functionality. The following table presents the time warping algorithm used by this option:

Present Year	Mapped to Past Year
2000, 2006, 2017	1995
2001, 2007, 2018	1990
2002, 2013, 2019	1991

2003, 2014	1997
*2004	1999, 1998
2005, 2011	1994
2008	1980
2009, 2015	1998
2010	1999
2012	1984
2016	1988
2020	1992
2021	1999

\* 1/1/2004 thru 2/28/2004 are mapped to 1998 calendar year,  
 2/29/2004 is mapped to 3/1/1998 (matches day of the week),  
 3 /1/2004 thru 12/31/2004 are mapped to 1999 calendar year.


- 3.) An OPC client can use these tags to import a CSV file into the OPC90 database. The CSV file can only contain new blocks to be added within an existing device. Lines that reference existing blocks or new devices will be ignored. The file to be imported is specified by the IMPORT\_FILE tag. The default directory of C:\Program Files\OPC90 Server\CSV will be assumed unless otherwise specified with the file name. Setting the IMPORT\_READ tag will request the CSV file to be imported. OPC90 will reset this flag when the import has been completed. The result of the import will be written to the IMPORT\_RESULT tag. A log file that summarizes the result of the import will also be generated. It is stored in the same directory as the CSV file imported. It has the same name as the CSV file but with a ".LOG" extension. The log file is a standard text file that can be viewed using NOTEPAD. After completion of the import, the OPC90 database is automatically saved.

### 7.8.1 SCSI Communication

The DEVICE definition block supports SCSI communication with the INICI03 and INICI13A. When a device block is configured, OPC90 will automatically detect the currently installed SCSI bus adapter(s) and search each adapter bus for INICI03 / INICI13A devices. Detected devices will appear in the primary and secondary port list boxes that are part of the DEVICE block properties. They are listed as devices like S0001:, S0002: etc. The 'S' indicates SCSI followed by the adapter card number, adapter bus number, adapter logical unit number and INICI03 SCSI target ID. Note that the target ID is the value set by switches on the Bailey INICI03 INICI13A module or INICI13A INICI13A module. The INICI03 or INICI13A must be powered up, configured to utilize its SCSI communication channel and attached to a PC SCSI host adapter card.

Just about any PC SCSI card that supports external connection to the INICI03 IMMPIO1 module's 50 pin SCSI socket or INICI13A SCSI 50 pin high density socket should work ok. Of course the SCSI card driver (provided by SCSI card manufacturer must also be installed).

Here is a table of SCS cards and supported operating systems that are being used with the Bailey INICI03 and INICI13A interfaces.

SCSI Card	PC Bus Type	Win 2000	Win XP	2003 Server	2008 Server	Server 2008 R2	Win 7
Adaptec 2915/2930LP	PCI		√ (32)				
Adaptec ASC-2930LPE	PCI Express		√ (32) 				
Adaptec AHA-2940AU	PCI	√ (32)	√ (32)				
LSI Logic LSI20320IE	PCI Express		√ (32)	√ (32)			
Adaptec 29320LPE	PCI Express			√ (32)	√ (32 & 64)	√ (32 & 64)	√ (32 & 64)
Adaptec 29160LP	PCI			√ (32)	√ (32 & 64)	√ (32 & 64)	√ (32 & 64)
Adaptec 29320ALP-R	PCI-X			√ (32 & 64)	√ (32 & 64)	√ (32 & 64)	√ (32 & 64)
HP SC11Xe	PCI Express			√ (32 & 64)	√ (32)		

A single SCSI card must be dedicated to communication with a single ABB Bailey SCSI CIU. Do not use the SCSI card to also communicate with other SCSI type devices such as hard disks, scanners, etc.

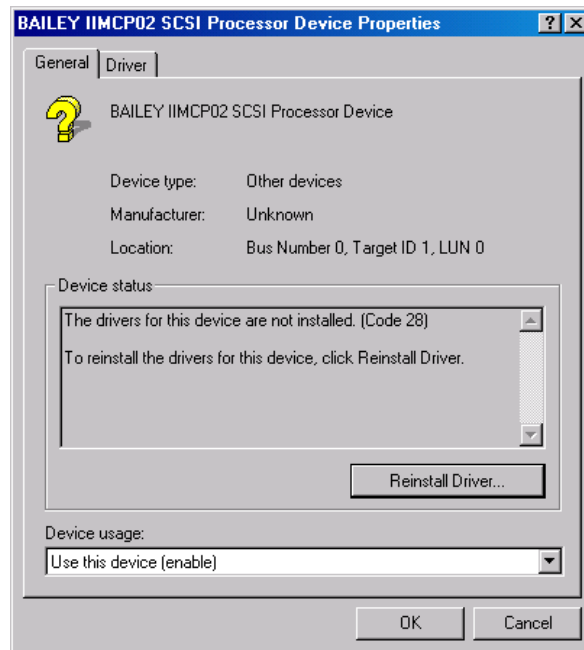
Adjustment to the SCSI BIOS might be necessary. The BIOS setup can be configured when the PC is booting. Following are general guidelines for typical settings. Depending on the type of SCSI card, all of these settings or the terminology might be different.

- \* SCSI Parity Checking: Enabled
- \* Host Adapter SCSI Termination: Automatic
- \* Initiate Sync Negotiation: Yes (IDs 0 - 7)
- \* Max Sync Transfer Rate: 20 MB/Sec (IDs 0 - 7)
- \* Enable Disconnect: Yes (IDs 0 - 7)
- \* Send Start Unit Command: No (IDs 0 - 7)
- \* BIOS Multiple LUN Support: No (IDs 0 - 7)
- \* Include in bios scan Yes (IDs 0 - 7)
- \* Plug and Play SCAM support: Disabled
- \* Reset SCSI Bus at IC Init: Disabled
- \* Extended bios Translate for DOS ICs: Enabled
- \* BIOS support for Int 13 extensions: Enabled

Depending on the operating system, OPC90 utilizes the Advanced SCSI Programming Interface (ASPI) or SCSI Pass Through Interface (SPTI) to communicate with the INICI03 or INICI13A via the host controller driver. If your system does not have ASPI it can be loaded by double clicking on the ASPI32.EXE file installed in the same directory as OPC90. Do NOT install ASPI32.EXE with Windows 2000, Windows ME, Windows XP, Easy CD Creator 4.x, or with Windows Media Player 7.0. If you have one of these applications (or operating systems), you will be using a different ASPI layer that will conflict with the one provided in this file.

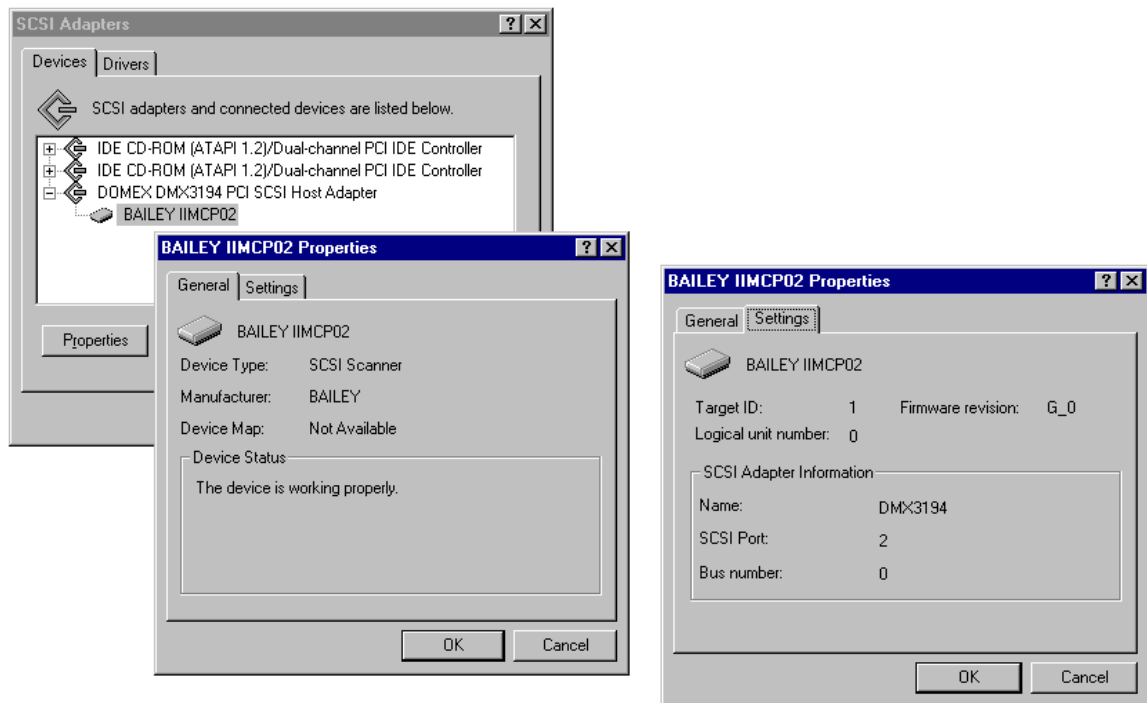
The only other driver that needs to be loaded is the standard manufacturer driver provided with the SCSI card or the resident driver recommended by Windows. It is not necessary to load the Bailey semAPI driver. Note that for some plug and play systems, the INICI03 or INICI13A will be listed as "Another Device" when the system is first booted up. This is normal. Accept it as another device and request it to be used (enabled) even

though a specific driver for that device has not been loaded. The properties for this other device will list the INICI03 as follows:



Notice that the INICI03 and INICI13A identifies itself to the SCSI adapter card as “BAILEY IIMCP02 SCSI Processor Device”. Again this is normal. You do not need to reinstall a driver for this device type. Also notice the location of the device is given by the above dialog. For this example the OPC90 will assigned “S0001:” as the device identity.

For systems that don’t support plug and play (default NT installations), the INICI03 or INICI13A device will appear under control panel, SCSI Adapters as follows:



Notice that the INICI03 or INICI13A identifies itself to the SCSI adapter card as “BAILEY IIMCP02 SCSI Scanner” This is normal. The location of the device is given in the settings property dialog. For this example OPC90 will assigned “S2001:” as the device identity.

## 7.8.2 TCP Communication

The DEVICE definition block supports TCP/IP communication with ABB IET800 (see Using OPC90 with IET800) or Lantronix Serial to Ethernet model UDS2100 device. Typically the Lantronix serial port redirector software is used to setup COM ports that appear as normal COM ports but behind the scene are mapped to the Lantronix device’s IP address and port number at that address. Communication using these COM ports is channeled through the Lantronix redirector driver.

The DEVICE block TCP ports can be used instead of the Lantronix redirected COM ports. This will bypass the communication being channeled through the Lantronix driver and instead OPC90 will setup a direct TCP socket connection with the Lantronix device. The DEVICE block supports definition of up to 16 TCP ports. These ports are referenced as TCP\_1P through TCP\_8P and TCP\_1S through TCP\_8S with the P signifying “Primary” and S “Secondary”.

The intent of this range of TCP ports is for organizational purposes only. Typically the first DEVICE block would be assigned TCP\_1P and TCP\_1S (if using redundant channel setup). The second DEVICE block, TCP\_2P and TCP\_2S and so on. In actuality any TCP port can be assigned in any order but just must be unique amongst all configured DEVICE blocks.

The list of TCP ports can be selected in the DEVICE block properties dialog and are found within the Primary and Secondary port list boxes. They appear after the list of 128 COM ports and any detected SCSI ports. The idea is to associate the TCP\_xP ports with the Primary port and TCP\_xS ports with Secondary port (when Redundant Channel or Dual Interface communication schemes are being used).

When a TCP port is initially selected it will not have an IP address and port number defined yet. The definition will look like this:

The screenshot shows the 'Device Properties' dialog box. The 'Name' field contains 'CIU' and the 'Description' field contains 'Infi 90 CIU'. Under the 'COMMUNICATION PARAMETERS' section, the 'Scheme' is set to 'Redundant Channel'. The 'Primary Port' section shows 'TCP\_1P:' selected from a dropdown, with empty text boxes for 'IP address here' and 'Port ?'. The 'Secondary Port' section shows 'TCP\_1S:' selected from a dropdown, with empty text boxes for 'IP address here' and 'Port ?'.

When setting up a Lantronix device, part of that setup includes assigning an IP address to it. By definition, the Lantronix first port number at that IP address will be 10001. This information is available from the Lantronix DeviceInstaller and / or CPR Manager software. When setting up the Lantronix device make sure to define the default baud rate and serial communication settings to match the settings of the CIU it is being cabled with. The recommended settings are 19200 baud, 8 data bits, 1 stop bit, no parity.

Define the Lantronix device IP and port number in the OPC90 DEVICE block Primary Port and Secondary Port (if using dual port configuration) property area. It will look like this:

The screenshot shows the 'Device Properties' dialog box with the communication parameters filled in. The 'Scheme' remains 'Redundant Channel'. The 'Primary Port' section now shows 'TCP\_1P:' selected, with '172.24.210.77' in the 'IP address here' box and '10001' in the 'Port ?' box. The 'Secondary Port' section shows 'TCP\_1S:' selected, with '172.24.210.77' in the 'IP address here' box and '10002' in the 'Port ?' box.

Note this example is using a CIU that supports two serial ports so the redundant channel scheme is shown. Most installations only use the Single Interface scheme so only the Primary TCP port would need to be defined for that case. Definition of this information must be done on each PC running OPC90. The TCP IP addresses and port numbers are saved in the registry and not in the OPC90 configuration file.

Save the configuration and put OPC90 into Monitor mode (View | Monitor) to verify it can communicate with the Lantronix Ethernet to Serial device. A quick check of the OPC90 DEVICE block faceplate will indicate whether or not successful communication is occurring and will show the actual IP and port number for the selected TCP port.

Note the DEVICE block support of direct TCP communication has been validated with Lantronix UDS2100 model device. It may work with other manufacturer Ethernet to Serial devices that support IP and port number addresses but the official device RoviSys claims support for is Lantronix.

### 7.8.3 Device Communication Faceplate

The DEVICE block includes a faceplate that summarizes important communication information associated with the device block. To display the faceplate, right click on the DEVICE block and select "Faceplate". The following faceplate is displayed:

**CIU Communication Information**

Symphony Plus IET800 Serial Connection

Mon Aug 6 14:28:27 2018

**STATUS**

Local	Shadow
Good P 192.168.123.22:10001	Unused Primary
Good S 192.168.123.22:10002	Unused Secondary

ONLINE, Unlimited V8.5.1

**ADDRESSES**

	Type	Ring	Node	Module
Local	IET800 A.6	1	22	2
Shadow	Unused	Unused	Unused	
Time Sync Master	Time Sync DCS	1	22	
Time Sync Master Accuracy Rating		12		

**STATISTICS**

	Totals	Rate (per second)
All Messages	245990	23
Exception Reports	2295427	438
Poll Messages	148740	13
Poll Block Output Values	1161068	118
Block Spec Read Cycles	126	
Exception Report Specs	2356	
Configured Blocks	12564	
Messages Queued	21	
Error Responses	9	

**Turbo**

Total	191
Active	177
Stale	0

Note that the “Time Sync DCS” button shown on this faceplate only appears if the Device block has been setup to participate in time syncing. It will appear gray while waiting for the proper conditions needed to request a DCS time sync to occur. The CIU must be communicating, not in standby operation, and the current time sync master has been determined. This is indicated by a non-zero node number appearing for the current time sync master. The current time sync master accuracy rating is also shown. Note that if the accuracy setting configured in the DEVICE block properties is less than the current time sync master accuracy rating, the CIU associated with this OPC90 cannot become time sync master.

The faceplate shows the total number of error responses received from the CIU. The error response button can be clicked to generate a snapshot report of all current error counters and command counters since startup of OPC90. This report provides context specific suggestions based on the presence of particular errors as can be seen from the following example.

-----  
 Nak report generated Wed Nov 18 08:18:26 2009  
 -----

Block--> CIU  
 Port---> S3000:  
 Device-> INICI03 G.3

<u>Total</u>	<u>Nak Code</u>	<u>Serious</u>	<u>Nak Description</u>
652180	0	No	Command complete
472726	1	No	Keyed command is being processed
1	105	No	Undefined block number - block is valid but not configured
1	110	No	Module not responding

Notes:

Nak codes 0 and 1 are actually positive acknowledgements and do not indicate an error condition.

Nak codes 100 - 112 are related to module configuration data. Their existence does not indicate a serious problem but could be a bad database configuration or a problem with ABB Bailey controllers.

Remember that most Nak codes are transient and recoverable.

No serious Naks have been detected.

-----  
 Commands sent report  
 -----

<u>Total</u>	<u>Cmd Code</u>	<u>Command Description</u>
--------------	-----------------	----------------------------



5	1	Establish point
1	10	Output value
441291	13	Read block
1	19	CIU restart
12430	20	Read block output
472724	25	Dequeue keyed command
11157	27	Demand module status
56213	34	Read work flag
673	43	Read system time-date
3367	49	Establish and connect point
332	60	Set system time-date
123422	63	Read data exception reports
346	66	Read data exception report specifications
1	69	Read environment
1	101	CIU online
2903	113	Extended read block
3	126	Extended support inquiry

## 7.9 Digital Input Loop (DIL)

The DIL OPC90 block is used to retrieve the exception reported output from Bailey Digital Output / Loop blocks (function code 45) and Digital Output / Loop With Deadband (function code 67). It can also be used to retrieve digital inputs defined within Bailey Logic Master Modules Group I/O definitions.

Restrictions: This OPC90 block can be utilized with all Bailey interface types except serial port module (SPM & CPM02).

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey module address.
ADDR_BLOCK	VT_I2	Config/Read	Bailey block address.
OUTLSD0	VT_BSTR	Config/Read	Output state zero logic state descriptor.
OUTLSD1	VT_BSTR	Config/Read	Output state one logic state descriptor.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
ALM_HOLD_ACT	VT_BOOL	Read	Alarm hold logic active indicator (see note 1).
ALM_HOLD_TIME	VT_I4	Read/Write	Alarm hold time in milliseconds (see note 1).
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey values (0-good, 1-bad).
DISC_LIM	VT_UI2	Read	Discrete alarm state.
DISC_ACT	VT_BOOL	Read	Alarm active indicator.
OUT	VT_BOOL	Read	Bailey discrete output value.
OUT_TEXT	VT_BSTR	Read	Bailey discrete output value as a string.

### Notes:

- 1.) Alarm hold logic holds the DISC\_ACT alarm indicator after the alarm condition no longer exists. The ALM\_HOLD\_ACT tag indicates a hold condition is in effect when TRUE (1). The duration of the hold is defined by ALM\_HOLD\_TIME expressed in milliseconds. When the block OUT value is no longer in the alarm state the alarm hold is cleared when the ALM\_HOLD\_TIME has transpired. The ALM\_HOLD\_TIME tag is read / write which means it can be changed in real time from the OPC client. Setting it to zero disables the hold feature.

## 7.10 ➡ Digital Output Loop (DOL)

The DOL OPC90 block is used to send an exception reported output generated by updates to the block input tag (IN). *Note that this block does not received data from a DOL (FC 45) block running in a Bailey controller. Use the OPC90 DIL block for that purpose.*

The exception reports generated by this block can be received by a Bailey console digital tag, Bailey Digital Input / Loop blocks (function code 42) and Digital Input / Inifnet blocks (function code 122). The OPC90 DOL block will automatically generate the appropriate quality, and alarming status based on the value presented at the block input.

Restrictions: This OPC90 block can be utilized with all Bailey interface types except serial port module (SPM & CPM02) and computer interface command series (CIC).

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey CIU ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey CIU node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey CIU module address.
ADDR_BLOCK	VT_I2	Config/Read	Block number to establish this point at within the Bailey interface (see note 1).
MAX_TIME	VT_I4	Config/Read	If IN never changes, it will be reported at the maximum time interval (seconds) defined by this attribute.
DISC_LIM	VT_BOOL	Config/Read	The setting for the Bailey alarm limit used to derive the alarm condition (see note 2).
INIT_VALUE	VT_BOOL	Config/Read	The IN value used to send initially when MAX_TIME is exceeded if an OPC client has not written to it.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
IN	VT_BOOL	Read/Write	Discrete input value to be exception reported to Bailey.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read/Write	Current quality (see note 3) of Bailey values (0-good, 1-bad).
DISC_ACT	VT_BOOL	Read	Alarm active indicator.
OUT	VT_BOOL	Read	Last discrete input value reported to Bailey.

Notes:

- 1.) Make sure the block number attribute is unique with respect to the other other DOL, AOL, ODD, OMSDD, ORCM, ORMC, ORMSC and OSTN OPC90 blocks associated with the same Bailey Interface Definition block. The block number attribute must also be defined within the range of 1 to maximum number of allowed outputs set up within the associated OPC90 Interface DEVICE block. The Bailey system will receive data from this block at the ring and node address of the Bailey CIU interface, module two and block number defined by the BLOCK attribute.

- 2.) This attribute is used to setup output alarming states. A value of 0 indicates flag alarm when output state is zero. A value of 1 indicates flag alarm when output state is one. A value of 2 indicates never flag an alarm.
- 3.) When the Device block "Set bad quality of max exception timeout" property is enabled, the QUALITY tag of this block will be set bad if writes to the block input(s) do not occur within the Exception Report Output Max Time setting of the block. When this occurs, bad quality will also be written to the CIU and therefore propagated to the users of the block data within the ABB Bailey system. The quality can also be set bad by writing a one (1) to this tag.

## 7.11 Harmony Analog Input (HAI)

The HAI OPC90 block is used to retrieve the exception reported output from Bailey Harmony Analog Input (function code 222) blocks.

Restrictions: This OPC block is only valid for Bailey interface types INICI03, INICI12 and INICI13.

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey module address.
ADDR_BLOCK	VT_I2	Config/Read	Bailey block address.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
CHAN_NAME	VT_BSTR	Read	Channel name assigned to this point
CHAN_TYPE	VT_I2	Read	Channel type code
CHAN_STATE	VT_BOOL	Read	Channel state (0-Ok, 1-Open, 2-Short, 3-Overdrive)
ALARM_CODE	VT_I2	Read	Alarm code
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey values (0-good, 1-bad).
QUAL_PROP	VT_BOOL	Read	Propogated Quality (0-Good, 1-Bad)
OUT	VT_R4	Read	Current output value from Bailey.
HIGH	VT_R4	Read	High signal value in Engineering Units.
LOW	VT_R4	Read	Low signal value in Engineering Units.
SIG_CHANGE	VT_R4	Read	Significant change in % of range
EU	VT_I2	Read	Engineering Units Code.
EU_TEXT	VT_BSTR	Read	Engineering Units String.
HI_LIM	VT_R4	Read/Write	Alarm limit used to detect the high alarm condition. Writing to this value tunes the alarm limit of the associated Bailey block.
LO_LIM	VT_R4	Read/Write	Alarm limit used to detect the low alarm condition. Writing to this value tunes the alarm limit of the associated Bailey block.
HI_ACT	VT_BOOL	Read	High alarm active indicator.
LO_ACT	VT_BOOL	Read	Low alarm active indicator.
OVR_ENABLE	VT_BOOL	Read/Write	Override allowed (0-No, 1-Yes)
OVR_VALUE	VT_R4	Read/Write	Override value
OVR_STATUS	VT_BOOL	Read	Value overridden (0-No, 1-Yes)
STAT_INHIBT	VT_BOOL	Read/Write	Status inhibit (0-No, 1-Yes)
SUSPECT	VT_BOOL	Read	Value suspect (0-No, 1-Yes)
DRIVEN_LO	VT_BOOL	Read	Low overdriven value (0-No, 1-Yes)
DRIVEN_HI	VT_BOOL	Read	High overdriven value (0-No, 1-Yes)
CONFIG_ERR	VT_BOOL	Read	Configuration error exists (0-No, 1-Yes)
READBACK	VT_BOOL	Read	Readback status (0-Good, 1-Bad)
REFERENCE	VT_BOOL	Read	Reference status (0-Good, 1-Bad)
CALIBRATION	VT_BOOL	Read	Calibration status (0-Good, 1-Bad)
SIM_ENABLE	VT_BOOL	Read/Write	Simulation enable (0-No, 1-Yes)
SIM_BLOCK	VT_I2	Read	Simulation value block number

## 7.12 Harmony Analog Output (HAO)

The HAO OPC90 block is used to retrieve the exception reported output from Bailey Harmony Analog Output (function code 223) blocks.

Restrictions: This OPC block is only valid for Bailey interface types INICI03, INICI12 and INICI13.

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey module address.
ADDR_BLOCK	VT_I2	Config/Read	Bailey block address.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
CHAN_NAME	VT_BSTR	Read	Channel name assigned to this point
CHAN_TYPE	VT_I2	Read	Channel type code
CHAN_STATE	VT_BOOL	Read	Channel state (0-Ok, 1-Open, 2-Short, 3-Overdrive)
ALARM_CODE	VT_I2	Read	Alarm code
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey values (0-good, 1-bad).
QUAL_PROP	VT_BOOL	Read	Propogated Quality (0-Good, 1-Bad)
OUT	VT_R4	Read	Current output value from Bailey.
HIGH	VT_R4	Read	High signal value in Engineering Units.
LOW	VT_R4	Read	Low signal value in Engineering Units.
SIG_CHANGE	VT_R4	Read	Significant change in % of range
EU	VT_I2	Read	Engineering Units Code.
EU_TEXT	VT_BSTR	Read	Engineering Units String.
HI_LIM	VT_R4	Read/Write	Alarm limit used to detect the high alarm condition. Writing to this value tunes the alarm limit of the associated Bailey block.
LO_LIM	VT_R4	Read/Write	Alarm limit used to detect the low alarm condition. Writing to this value tunes the alarm limit of the associated Bailey block.
HI_ACT	VT_BOOL	Read	High alarm active indicator.
LO_ACT	VT_BOOL	Read	Low alarm active indicator.
OVR_ENABLE	VT_BOOL	Read/Write	Override allowed (0-No, 1-Yes)
OVR_VALUE	VT_R4	Read/Write	Override value
OVR_STATUS	VT_BOOL	Read	Value overridden (0-No, 1-Yes)
STAT_INHIBT	VT_BOOL	Read/Write	Status inhibit (0-No, 1-Yes)
SUSPECT	VT_BOOL	Read	Value suspect (0-No, 1-Yes)
DRIVEN_LO	VT_BOOL	Read	Low overdriven value (0-No, 1-Yes)
DRIVEN_HI	VT_BOOL	Read	High overdriven value (0-No, 1-Yes)
CONFIG_ERR	VT_BOOL	Read	Configuration error exists (0-No, 1-Yes)
READBACK	VT_BOOL	Read	Readback status (0-Good, 1-Bad)
REFERENCE	VT_BOOL	Read	Reference status (0-Good, 1-Bad)
CALIBRATION	VT_BOOL	Read	Calibration status (0-Good, 1-Bad)
SIM_ENABLE	VT_BOOL	Read/Write	Simulation enable (0-No, 1-Yes)
SIM_BLOCK	VT_I2	Read	Simulation value block number

### 7.13 Harmony Digital Input (HDI)

The HDI OPC90 block is used to retrieve the exception reported output from Bailey Harmony Digital Input (function code 224) blocks.

Restrictions: This OPC block is only valid for Bailey interface types INICI03, INICI12 and INICI13. This OPC block does not support the reception of Harmony digital input sequence of events data.

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey module address.
ADDR_BLOCK	VT_I2	Config/Read	Bailey block address.
OUTLSD0	VT_BSTR	Config/Read	Output state zero logic state descriptor.
OUTLSD1	VT_BSTR	Config/Read	Output state one logic state descriptor.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
CHAN_NAME	VT_BSTR	Read	Channel name assigned to this point
CHAN_STATE	VT_BOOL	Read	Channel state (0-Ok, 1-Open, 2-Short, 3-Overdrive)
ALARM_CODE	VT_I2	Read	Alarm code
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey values (0-good, 1-bad).
QUAL_PROP	VT_BOOL	Read	Propogated Quality (0-Good, 1-Bad)
OUT	VT_BOOL	Read	Current output value from Bailey.
OUT_TEXT	VT_BSTR	Read	Discrete Output Value as Text
DISC_LIM	VT_UI2	Read	Discrete alarm state.
DISC_ACT	VT_BOOL	Read	Alarm active indicator.
OVR_ENABLE	VT_BOOL	Read/Write	Override allowed (0-No, 1-Yes)
OVR_VALUE	VT_BOOL	Read/Write	Override value
OVR_STATUS	VT_BOOL	Read	Value overridden (0-No, 1-Yes)
STAT_INHIBT	VT_BOOL	Read/Write	Status inhibit (0-No, 1-Yes)
SUSPECT	VT_BOOL	Read	Value suspect (0-No, 1-Yes)
CONFIG_ERR	VT_BOOL	Read	Configuration error exists (0-No, 1-Yes)
READBACK	VT_BOOL	Read	Readback status (0-Good, 1-Bad)
SIM_ENABLE	VT_BOOL	Read/Write	Simulation enable (0-No, 1-Yes)
SIM_BLOCK	VT_I2	Read	Simulation value block number

## 7.14 Harmony Digital Output (HDO)

The HDO OPC90 block is used to retrieve the exception reported output from Bailey Harmony Digital Output (function code 225) blocks.

Restrictions: This OPC block is only valid for Bailey interface types INICI03, INICI12 and INICI13.

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey module address.
ADDR_BLOCK	VT_I2	Config/Read	Bailey block address.
OUTLSD0	VT_BSTR	Config/Read	Output state zero logic state descriptor.
OUTLSD1	VT_BSTR	Config/Read	Output state one logic state descriptor.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
CHAN_NAME	VT_BSTR	Read	Channel name assigned to this point
CHAN_STATE	VT_BOOL	Read	Channel state (0-Ok, 1-Open, 2-Short, 3-Overdrive)
ALARM_CODE	VT_I2	Read	Alarm code
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey values (0-good, 1-bad).
QUAL_PROP	VT_BOOL	Read	Propogated Quality (0-Good, 1-Bad)
OUT	VT_BOOL	Read	Current output value from Bailey.
OUT_TEXT	VT_BSTR	Read	Discrete Output Value as Text
DISC_LIM	VT_UI2	Read	Discrete alarm state.
DISC_ACT	VT_BOOL	Read	Alarm active indicator.
OVR_ENABLE	VT_BOOL	Read/Write	Override allowed (0-No, 1-Yes)
OVR_VALUE	VT_BOOL	Read/Write	Override value
OVR_STATUS	VT_BOOL	Read	Value overridden (0-No, 1-Yes)
STAT_INHIBT	VT_BOOL	Read/Write	Status inhibit (0-No, 1-Yes)
SUSPECT	VT_BOOL	Read	Value suspect (0-No, 1-Yes)
CONFIG_ERR	VT_BOOL	Read	Configuration error exists (0-No, 1-Yes)
READBACK	VT_BOOL	Read	Readback status (0-Good, 1-Bad)
SIM_ENABLE	VT_BOOL	Read/Write	Simulation enable (0-No, 1-Yes)
SIM_BLOCK	VT_I2	Read	Simulation value block number



## 7.15 Module Status (MODSTAT)

The MODSTAT OPC90 block is used to retrieve module status summary information and detailed module problem reports. This block is designed to work with all Bailey node and module types.

Restrictions: This OPC90 block can be utilized with all Bailey interface types except serial port module (SPM & CPM02).

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey ring address (note 1).
ADDR_NODE	VT_I2	Config/Read	Bailey node address (note 1).
ADDR_MODULE	VT_I2	Config/Read	Bailey module address (note 1).
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
COLUMNS	VT_I2	Read/Write	Number of columns (characters) per line to use when generating problem report text. Values from 50 to 130 are allowed.
READ	VT_BOOL	Read/Write	Set this attribute to request module problem reports to be read for the Bailey module status defined for this block. The problem reports are written to the LINE01 to LINE50 attributes. The driver will reset this attribute when the read request is completed.
NEXT	VT_BOOL	Read/Write	Set this attribute to request a read of the next group of problem reports. The problem reports are written to the LINE01 to LINE50 attributes. The driver will reset this attribute when the next problem report read request is completed.
LINES	VT_I2	Read/Write	Number of lines to utilize when generating problem report text. Values from 2 to 50 are allowed.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey values (0-good, 1-bad).
OUT	VT_BOOL	Read	Indicates whether or not the Bailey module is currently experiencing any errors. A value of zero indicates no errors. A value of one indicates one or more errors. They can be determined by examining the STATUS_TEXT attribute and requesting problem reports.
LINE01 to LINE50	VT_BSTR	Read	Requested problem report text strings are written to these attributes.
STATUS_TEXT	VT_BSTR	Read	Overview status of the addressed Bailey module (see note 2).
STAT_BYTE01 to STAT_BYTE16	VT_I2	Read	Bailey module status bytes. Only the first 5 are valid for Command Series and Network

			90 systems. All sixteen are valid for Infi 90 systems.
STAT_BYTE01_BIT0 to STAT_BYTE16_BIT7	VT_BOOL	Read	Bailey module status bytes broken apart into individual bits. All eight bits are presented for all sixteen module status bytes. Bit 0 is the least significant bit.

Notes:

- 1.) When retrieving the module status of a CIU interface special attention must be given to the settings of ring, node and module. For all CIU interfaces except the CIU01, the ring must match the ring address of the CIU, node must match the node address of the CIU and **module must be set to a value of two**. For the CIU01 interface, ring and node must be set to a value of zero and module to a value of two. Failure to follow these rules will result in improper or no module status returned for the CIU interface.
- 2.) The status of the Bailey module returned as a string in STATUS\_TEXT tag. The following format "<Type>, <State>, <Errors>" where type indicates the module type (AMM, MFC, COM, MFP, BRC, etc), state indicates its current state of operation (Execute, Configure, Error) and errors are a summary of any current errors it is experiencing. The returned status string may also contain the following codes based on the particular module type.

FTX – first time in execute

BCKBAD – backup controller is bad

MEMFULL – memory is full

RIOBAD – remote I/O (values brought in from outside controller) with bad quality

LIOBAD – local I/O (values read from inside controller) with bad quality

BCKDIFF – backup controller has different configuration (happens during online configuration)

SEGALM – segment blocks are overrunning targeted run time

NVMBAD – non-volatile storage is bad

NOCFG – controller has no configuration

STNBAD – hard controller station(s) is bad

PCUOFF – one or more PCU nodes offline (doesn't mean the OPC90 CIU node is offline)

RINGOFF – one or more rings offline (doesn't mean the OPC90 CIU ring is offline)

MEMFULL – memory is full

ENVBAD – environment is bad (power supply or high temperature)

BACKUPBAD – PCU backup access module is bad

LIMBAD – loop interface module bad (Network 90 systems)

RX1BAD – receiver channel #1 bad (Network 90 systems)

RX0BAD – receiver channel #0 bad (Network 90 systems)

TX1BAD – transmitter channel #1 bad (Network 90 systems)

TX0BAD – transmitter channel #0 bad (Network 90 systems)

## 7.16 Multi-State Device Driver (MSDD)

The MSDD OPC90 block is used to retrieve and control the exception reported output from Bailey Multi-State Device Driver blocks (function code 129).

Restrictions: This OPC90 block can be utilized with all Bailey interface types except serial port module (SPM & CPM02).

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey module address.
ADDR_BLOCK	VT_I2	Config/Read	Bailey block address.
F1LSD0	VT_BSTR	Config/Read	Feedback 1 state zero logic state descriptor.
F1LSD1	VT_BSTR	Config/Read	Feedback 1 state one logic state descriptor.
F2LSD0	VT_BSTR	Config/Read	Feedback 2 state zero logic state descriptor.
F2LSD1	VT_BSTR	Config/Read	Feedback 2 state one logic state descriptor.
F3LSD0	VT_BSTR	Config/Read	Feedback 3 state zero logic state descriptor.
F3LSD1	VT_BSTR	Config/Read	Feedback 3 state one logic state descriptor.
F4LSD0	VT_BSTR	Config/Read	Feedback 4 state zero logic state descriptor.
F4LSD1	VT_BSTR	Config/Read	Feedback 4 state one logic state descriptor.
GSLSD0	VT_BSTR	Config/Read	Good state zero logic state descriptor.
GSLSD1	VT_BSTR	Config/Read	Good state one logic state descriptor.
GSLSD2	VT_BSTR	Config/Read	Good state two logic state descriptor.
GSLSD3	VT_BSTR	Config/Read	Good state three logic state descriptor.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
FACE_TYPE	VT_I2	Read	Faceplate type (MSDD, S18).
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey values (0-good, 1-bad).
DISC_ACT	VT_BOOL	Read	Alarm active indicator.
OUT	VT_BOOL	Read	Discrete output value of Bailey MSDD first control output signal.
MODE	VT_I2	Read/Write	The mode of the Bailey block. MODE can be: 0 – Manual 1 – Auto
RED_CMD	VT_BSTR	Read/Write	Red tag command (see red tag users section).
RED_TAG	VT_BOOL	Read	Red tag indicator (0 – no, 1 – yes).
RED_USERS	VT_BSTR	Read	Red tag users (current user names or codes).
F1	VT_BOOL	Read	State of feedback number 1.
F1_TEXT	VT_BSTR	Read	State of feedback number 1 as a string.
F2	VT_BOOL	Read	State of feedback number 2.
F2_TEXT	VT_BSTR	Read	State of feedback number 2 as a string.
F3	VT_BOOL	Read	State of feedback number 3.
F3_TEXT	VT_BSTR	Read	State of feedback number 3 as a string.
F4	VT_BOOL	Read	State of feedback number 4.

F4_TEXT	VT_BSTR	Read	State of feedback number 4 as a string.
GOOD_STATE	VT_I4	Read	Good state (see note 1).
GS_TEXT	VT_BSTR	Read	Good state as a string.
REQ_STATE	VT_I4	Read/Write	Requested state (see note 1).
RS_TEXT	VT_BSTR	Read/Write	Requested state as a string (see note 2).
TRAVEL	VT_BOOL	Read	Travel indicator (see note 3).
OVR_CONTROL	VT_BOOL	Read	Override control indicator.
OVR_STATUS	VT_BOOL	Read	Override status indicator.
DEFAULT_MASK	VT_I2	Read	Default state output mask (MSDD, S7).
OUT_MASK1	VT_I2	Read	State 1 output mask (MSDD, S8).
OUT_MASK2	VT_I2	Read	State 2 output mask (MSDD, S9).
OUT_MASK3	VT_I2	Read	State 3 output mask (MSDD, S10).
FB_MASK1	VT_I2	Read/Write	Feedback mask for state 1 (MSDD, S11).
FB_MASK2	VT_I2	Read/Write	Feedback mask for state 2 (MSDD, S12).
FB_MASK3	VT_I2	Read/Write	Feedback mask for state 3 (MSDD, S13).
NEXT1	VT_I2	Read/Write	Next allowable state(s) from state 1 (MSDD, S19).
NEXT2	VT_I2	Read/Write	Next allowable state(s) from state 2 (MSDD, S20).
NEXT3	VT_I2	Read/Write	Next allowable state(s) from state 3 (MSDD, S21).
FB_WAIT	VT_R4	Read/Write	Feedback waiting time in seconds (MSDD, S16).
FAULT_WAIT	VT_R4	Read/Write	Fault waiting time in seconds (MSDD, S17).
PULSE_WAIT	VT_R4	Read/Write	Length of pulsed outputs in seconds (MSDD, S22).
MAN_PERMIT	VT_I1	Read/Write	Manual mode permit flag (1 – yes) (MSDD, S15).
OVERRIDES	VT_I2	Read/Write	Current status and control override definitions (MSDD, S14).

#### Notes:

- 1.) The GOOD\_STATE and REQ\_STATE (requested state) attributes are used to view current operating state of the block and command its requested state. Valid values are: (zero = default, one = state 1, two = state 2, and three = state 3). Writing any of these values to REQ\_STATE commands the MSDD to that associated state. If the controller MSDD block has pulsed outputs enabled (S22 non-zero), the REQ\_STATE will automatically reset to the default state (0) when the pulsed outputs return to the default state. This feature can be disabled by turning off the OPC90 Device block “MSDD Pulse Out Handling” property.
- 2.) Text strings can be written to this tag to control the MSDD requested state. Valid text strings are the GSLSD0, GSLSD1, GSLSD2 and GSLSD3 settings along with “Default”, “0”, “1”, “2”, “3”, “State 0”, “State 1”, “State 2”, “State 3”, “State0”, “State1”, “State2” and “State3”. Writing any of these strings is case insensitive.
- 3.) Travel indicator is not actually received from Bailey but composed by the driver for display indication purposes. It is set for the interval of time between commanding a new requested state and waiting for the good state to arrive at the requested state while no alarm has been flagged by the Bailey MSDD block.

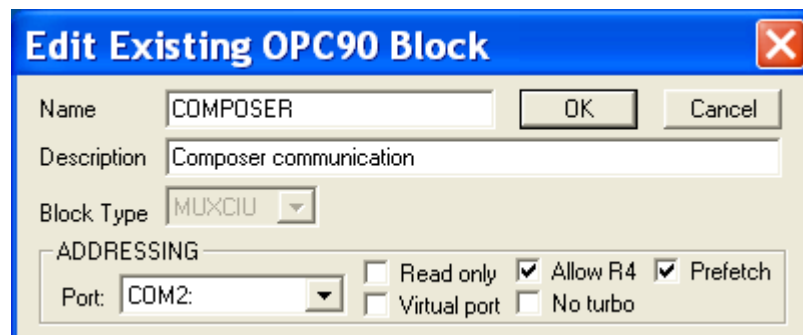
## 7.17 Multiplex CIU (MUXCIU)

The MUXCIU OPC90 block allows a configuration utility program, requiring communication access to the Bailey system, to share the CIU device associated with the OPC90 DEVICE block. This block does not support programs that need to establish points in the CIU. Use this block to allow the G.Michaels DBDOC, Bailey CADEWS or Composer software to communicate with the Bailey system via the OPC90 DEVICE block driver.

To accomplish this task, connect a RS232 serial cable between the COM port specified by the MUXCIU PORT attribute and the COM port to which the configuration utility program expects the Bailey CIU interface to be attached. Note that this connection must be made with a null modem cable. The MUXCIU block receives CIU commands sent by the configuration utility program via the assigned serial port. It channels those commands to the Bailey system through the OPC90 DEVICE block driver. The CIU reply is then returned to the configuration utility program via the assigned serial port. All commands not associated with points established in the Bailey interface are supported.

Configuration of this block includes four options.

The “read only operation” option filters commands that make changes in the ABB Bailey DCS. This option can be enabled by clicking on the “Read only operation” as shown by the following block properties dialog.



This option should be enabled when using this block with DBDOC. Do not enable it when using with Bailey CADEWS or Composer software.

The “Allow R4” option allows polled block outputs to be read as real 4 values. Note that the Device block “Enhanced Analog Precision” must also be enabled to allow reading of real 4 values. If the configuration software associated with the MUXCIU block does not support real 4 values, this option should be disabled.

The “Virtual port connection” option enables virtual connections to this MUXCIU block. OPC90 includes a simple application programming interface DLL called OPC90VPort.DLL that allows applications such as the DBDOC CIUMON program to connect with OPC90 MUXCIU blocks virtually. The virtual connection can be local to the OPC90 PC or by remote PCs residing on the same network as the OPC90 PC.

The “No turbo” option can be used to turn off turbo polling of block output reads being channeled through the MUXCIU block. It is useful for disabling turbo for a specific MUXCIU while leaving it on for others and OPC90 in general (note the device block properties is used to enable turbo polling globally). An example reason for turning turbo off would be with Composer communicating through a MUXCIU block and the need to see alarm bits that are available from polling exception report block outputs. Since turbo polling does not include that information, it could be turned off so it is available in Composer.

The “Prefetch” option enables the MUXCIU to use parallel pre-fetch logic when it detects a module save or restore operation occurring. Parallel pre-fetch causes the MUXCIU block to request replies to commands it anticipates the attached software will be asking next. This allows module save and restore operations to occur at a maximum possible rate considering the double transmission nature of using the MUXCIU block. This option should be disabled when the COM port connection between the MUXCIU and attached software such as Composer are using the G. Michaels Consulting “Virtual Port to Virtual Port” feature.

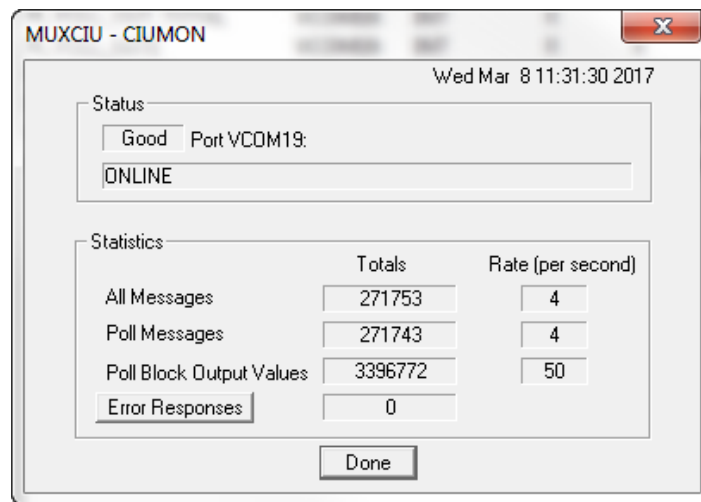
Commands that would be detrimental to the operation of OPC90 communication with the Bailey interface are handled directly by the MUXCIU block. For example, a common command that a configuration utility program typically sends upon initially starting up, is the CIU RESTART command. The MUXCIU block intercepts this command and does not actually send it to the OPC90 DEVICE driver. Instead it simply returns the response originally received when the OPC90 DEVICE block driver had previously sent the CIU RESTART command. The MUXCIU block has attributes to monitor communication health and statistics. Italicized names indicate attributes configured within the OPC90 Server, which cannot be connected to via OPC.

Restrictions: None.

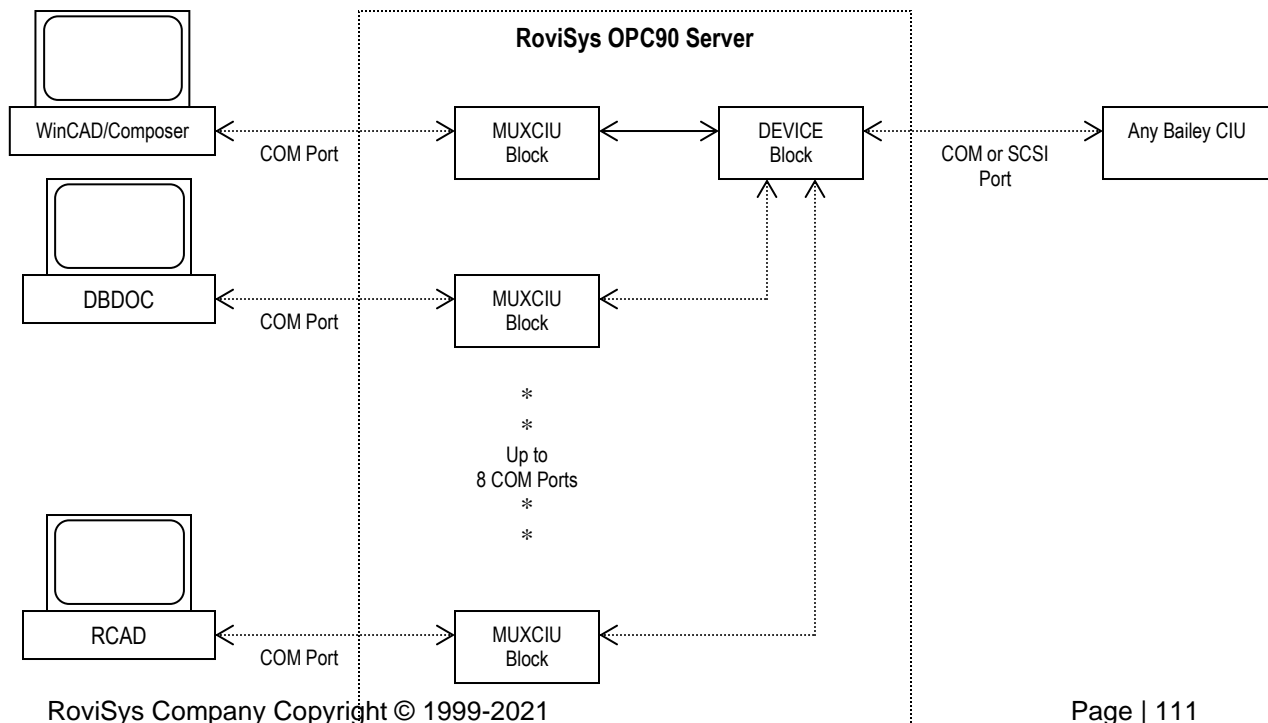
<b>TAGNAME</b>	<b>TYPE</b>	<b>ACCESS</b>	<b>DESCRIPTION</b>
<i>Port</i>	VT_BSTR	Config	Name of the COM port to be multiplexed as a CIU port.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
COMM_STATUS	VT_I4	Read	Communication channel status. A value of zero indicates good and one means bad.
MSG_TOTAL	VT_I4	Read	Running count of all messages exchanged with the external software.
MSG_RATE	VT_I4	Read	Rate (per second) of all messages exchanged with the external software.
POLL_TOTAL	VT_I4	Read	Running count of poll messages (those generating GMI activity) being exchanged with the external software.

POLL_RATE	VT_I4	Read	Rate (per second) of poll messages (those generating GMI activity) exchanged with the external software.
POLL_OUT_TOTAL	VT_I4	Read	Running count of block output values exchanged with the external software.
POLL_OUT_RATE	VT_I4	Read	Rate (per second) of block output values exchanged with the external software.
NAK_TOTAL	VT_I4	Read	Running count of total negative acknowledgments sent to the external software.

The MUXCIU block faceplate presents a nice snapshot of current runtime operation. Just double click on the block to view faceplate which looks as follows.



The following diagram shows the logical connections when utilizing the MUXCIU block.



The following table lists some of the Bailey interface commands supported by the MUXCIU block. Commands not supported will receive a “Command Not Supported” (NAK-3) response. When the “Read only operation” option is enabled, commands that cause changes in the ABB Bailey DCS will not be sent to the CIU and a “Command Not Supported” (NAK-3) response will be generated.

Command	Command Code	Response
RE-READ REPLY	9	Resends the last command response.
MODULE OPERATION	12	ACK/NAK code indicating requested result.
READ BLOCK	13	Returns requested block data.
READ NEXT BLOCK	14	Returns next block requested block data.
READ DEFAULT BLOCK	15	Returns default block data.
WRITE BLOCK	16	ACK/NAK code indicating requested result.
TUNE BLOCK	17	ACK/NAK code indicating requested result.
DELETE BLOCK	18	ACK/NAK code indicating requested result.
READ BLOCK OUTPUT	20	Returns value of requested block output.
READ PROBLEM REPORT	26	Returns requested problem report data.
DEMAND MODULE STATUS	27	Returns requested module status.
READ EXTENDED PROBLEM REPORT	46	Returns requested extended problem report data.
TREND DATA POLL	48	Returns requested trend block data.
CIU RESTART	19	Returns response from when OPC90 DEVICE block driver originally issuing this command.
READ WORK FLAG	34	Always return ACK and flags indicating no additional work is required.
ENVIRONMENT	69	Returns response from when OPC90 DEVICE block driver originally issuing this command.
READ SYSTEM TIME/DATE	43	Returns requested time/date response. Time Stamp and Wall Clock offset are always zero.
SET SYSTEM TIME/DATE	60	Always returns a ACK response.
DEQUEUE	25	Always returns a ACK response.
CIU CALLUP	41	Always returns a ACK response.
CIU HANGUP	42	Always returns a ACK response.
CIU ONLINE / OFFLINE	44	Always returns a ACK response.
CANCEL KEY COMMAND	50	Always returns a ACK response.
DEFINE SYSTEM NODE	61	Always returns a ACK response.

### 7.17.1 MUXCIU Virtual Port API

The OPC90VPort API allows third party applications that utilize the CIU for non-database type of activities to safely and easily share the CIU with OPC90 without the need for a physical connection. The third party application can be running on the same PC as OPC90 or on another PC that is on the same network as the OPC90 PC.

This API is implemented as a C++ class and contains three functions. These functions are, OPC90Connect(), OPC90CommandCIU() and OPC90Disconnect() .



These three functions are callable in the OPC90VPort.dll. See the OPC90Vport.h header file for details of calling each function. That header must be included in the third party applications development environment along with the OPC90VPort.lib file.

OPC90Connect() is used to establish a virtual connection with a specific MUXCIU block configured in the OPC90 database and setup for virtual communication. Once connected, the third party application is granted exclusive use of that block.

OPC90CommandCIU() is used to send CIU commands to the actual CIU via the MUXCIU block virtual connection. These commands are of the format of fully byte stuffed array that would normally be ready to send to the physical CIU by the third party application. The return from this function is the response for the command.

OPC90Disconnect() is used to give up the virtual connection with the MUXCIU block.

The OPC90VPortTester utility included with OPC90 uses this API and can be used to test a virtual port connection with a MUXCIU block. It is also a useful utility for general purpose polling of block outputs within a given ABB Bailey controller module.

Here is an example of connecting and disconnecting with a MUXCIU block. Note that an application should connect, do all of its communication using the OPC90CommandCIU() call and when finished disconnect.

```
#include      "OPC90VPort.h";
COPC90VPort  opc90;
WORD         result;
DWORD        sessionID;

// connect with the MUXCIU block
result = opc90.OPC90Connect( "OPC90PC",           // name of OPC90 PC
                             "BLOCKNAME",         // long tag name of MUXCIU block
                             30000,               // time out in 30000 milliseconds no activity
                             &sessionID );        // session ID needed by the other calls

// disconnect from the MUXCIU block
result = opc90.OPC90Disconnect( sessionID );
```

## 7.18 Network Interface Slave Monitor (NISMON)

The NISMON OPC90 block is used to retrieve general network interface slave (NIS) information, topology data, event and error counters, memory usage and various node performance statistics. NISMON is typically used to return topology data but also network diagnostic information to help find failing node hardware.

Each NISMON block automatically maintains a daily log in the OPC90 log directory. The block includes logic that constantly observes the received data looking for possible problems. If a potential problem is detected it is added to the NISMON daily log file. The 30 most recent log entries can also be viewed from the NISMON faceplate.

The NISMON block is typically used to return loop topology information. When reading of the topology is the only purpose for using this block only one needs to be configured per communication loop. All nodes in a communication loop share a node map with each other that defines the topology for the loop. Therefore, when the NISMON block is only being used to read the topology and the other data it returns is not required, configuration of only one block per loop is necessary. In this case it is recommended that all of the enable tags be turned off except topology enable. This will eliminate unnecessary data collection.

**Cautions:** This block can be useful as a diagnosis tool for INFI 90, Harmony and Symphony+ communication system. Be careful with the INTERVAL setting, especially when multiple NISMON blocks are configured. After completing an analysis of a node operational state, set INTERVAL to a higher value or zero to disable data collection if it is currently not needed. This block uses polling to get the data which means it does use a portion of the available polling bandwidth between OPC90 and the ABB Bailey CIU device and also the node it is addressed too.

**Restrictions:** This OPC90 block is valid for INFI 90, Harmony and Symphony+ systems. It can be utilized with Bailey interface CIU types that include an INICT03A or INICT13A module. The node to be monitored must have INNPM01 firmware revision C\_1 or later. Note that the OPC90 MODSTAT block can be used to determine the firmware revision level of the INNPM01 if it is not known. All INNPM22 firmware revisions are supported for nodes so equipped.

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/ Read/Write	Bailey PCU ring address (see note 1).
ADDR_NODE	VT_I2	Config/ Read/Write	Bailey PCU node address (see note 1).
INTERVAL	VT_UI4	Config/ Read/Write	How often to read the data in ms, 0-disable (see note 2).
GET_NOW	VT_BOOL	Read/Write	Set to request an immediate read of the enabled data. INTERVAL must also be non-zero.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.

ENABLE_NIS	VT_BOOL	Read/Write	Enable collection of NIS event and error counters (see note 2).
ENABLE_NPS	VT_BOOL	Read/Write	Enable collection of node performance statistics (see note 2).
ENABLE_MEM	VT_BOOL	Read/Write	Enable collection of module memory utilization (see note 2).
ENABLE_XRP	VT_BOOL	Read/Write	Enable collection of module exception statistics (see note 2).
ENABLE_MAP	VT_BOOL	Read/Write	Enable collection of loop topology map (see note 3).
ENABLE_SYNC	VT_BOOL	Read/Write	Enable collection of current time sync master ID (see note 2).
SAMPLES	VT_UI1	Read/Write	Number of samples for calculating moving averages.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of PCU node communication statistics (0-good, 1-bad).
FIRMWARE	VT_BSTR	Read	NIS Firmware revision.
SW1	VT_UI1	Read	PCU address switch setting.
SW2	VT_UI1	Read	Loop (ring) address switch setting.
SW3	VT_UI1	Read	Loop (ring) mode switch setting.
SW4	VT_UI1	Read	LED display / expander bus address switch setting.
<b>When ENABLE_MAP</b>			<b>Loop (Ring) Topology:</b>
TOPOLOGY_INTERVAL	VT_UI4	Read/Write	How often to read the topology data in ms, 0-disable (see note 3).
TOPOLOGY	VT_BSTR	Read	Topology of the currently addressed loop (see note 3).
<b>When ENABLE_NIS</b>			<b>Resettable NIS Event and Error Counters:</b>
TIME_IRQ	VT_UI4	Read/Write	Time (milliseconds) since last module or command reset. Write 0 to this tag to reset event and error counters.
CHAN1_RX_ERR	VT_UI4	Read	Channel 1 receive errors.
CHAN2_RX_ERR	VT_UI4	Read	Channel 2 receive errors.
TX_ERR	VT_UI4	Read	Transmit errors.
LOST_RQ_OVERFLOW	VT_UI4	Read	Messages lost to receive queue overflow.
DMP_CIR_CNT_ERR	VT_UI4	Read	Messages dumped with circulation count errors.
DMP_TYPE_CNT_ERR	VT_UI4	Read	Messages dumped with message type or destination count errors.
DMP_SRC_STATE_ERR	VT_UI4	Read	Messages dumped with source state errors.
DMP_SRC_SEQ_ERR	VT_UI4	Read	Messages dumped with source sequence mismatch.
MULTI_CAST_RX	VT_UI4	Read	Multicast messages received excluding original.
MULTI_CAST_DEST_RX	VT_UI4	Read	Multicast destinations received.
TIME_SYNC_RX	VT_UI4	Read	Time sync messages received excluding original.

BROADCAST_RX	VT_UI4	Read	Broadcast messages received excluding original.
NIS_POLL_RX	VT_UI4	Read	NIS poll messages received excluding original.
NIS_POLL_ACK	VT_UI4	Read	NIS poll messages acknowledged by this node.
NIS_POLL_BUSY	VT_UI4	Read	NIS poll messages busy not acknowledged by this node.
MSG_TOTAL_TX	VT_UI4	Read	Total messages transmitted - total loop traffic.
MSG_TOTAL_RX_TX	VT_UI4	Read	Total messages received and forwarded by this node.
MSG_ORIGINATED	VT_UI4	Read	Total messages originated by this node including retries.
BYTES_TOTAL_RX_TX	VT_UI4	Read	Total bytes received and forwarded by this node.
XBUS_TO_NIS_MSG	VT_UI4	Read	Expander bus to NIS handshake messages.
XBUS_SIGNALS	VT_UI4	Read	Expander bus message to transmitt buffer signals.
XBUS_STAT_REQ	VT_UI4	Read	Expander bus PCU status requests.
XBUS_NIS_STAT_REQ	VT_UI4	Read	Expander bus NIS status requests.
XBUS_NIS_IRQ_BAD	VT_UI4	Read	Expander bus interrupts with invalid status.
ALIGN_TX	VT_UI4	Read	Re-alignment of transmit buffer due to invalid contents.
ALIGN_RX	VT_UI4	Read	Re-alignment of receive buffer.
ALIGN_STAT	VT_UI4	Read	Re-alignment of status buffer.
ALIGN_RX_REQ	VT_UI4	Read	Re-alignment of receive buffer requests.
ALIGN_STAT_REQ	VT_UI4	Read	Re-alignment of status buffer requests.
STAT_OFLOW_NODE	VT_UI4	Read	Status buffer message overflow node status.
READ_TSYNC_REQ	VT_UI4	Read	Read time sync buffer requests.
SPUR_IRQ7	VT_UI4	Read	Spurious level 7 interrupts.
SPUR_IRQ6	VT_UI4	Read	Spurious level 6 interrupts.
SPUR_IRQ5	VT_UI4	Read	Spurious level 5 interrupts.
LOOP_FAILS	VT_UI4	Read	Number of loop failures.
<b>When ENABLE_SYNC</b>			<b>Time Sync Master ID:</b>
SYNC_RING	VT_UI1	Read	Time sync master ring.
SYNC_NODE	VT_UI1	Read	Time sync master node.
SYNC_ACCURACY	VT_UI1	Read	Time sync master accuracy rating.
<b>When ENABLE_MEM</b>			<b>Module Memory Utilization:</b>
MEMORY_TOTAL	VT_UI4	Read	Total memory bytes.
MEMORY_UNUSED	VT_UI4	Read	Unused memory bytes.
MEMORY_TEMP	VT_UI4	Read	Temporary memory bytes.
MEMORY_FREE	VT_R4	Read	Percent memory free.
<b>When ENABLE_XRP</b>			<b>Resettable Module Exception Statistics:</b>
XRP_RESET_TIME	VT_UI4	Read/Write	Time (milliseconds) since last module reset or XRP statistics reset command. Write 0 to this tag to reset statistics time.
XRP_RX_PEAK	VT_UI4	Read	Exception receive rate peak (points / second).
XRP_TX_PEAK	VT_UI4	Read	Exception transmit rate peak (points / second).
XRP_RX_COUNT	VT_UI4	Read	Exception receive count (points).

XRP_TX_COUNT	VT_UI4	Read	Exception transmit count (points).
XRP_RX_PACK_COUNT	VT_UI4	Read	Exception receive packet count.
XRP_TX_PACK_COUNT	VT_UI4	Read	Exception transmit packet count.
XRP_RX_PACK_PCT	VT_UI1	Read	Exception receive pack percentage (see note 4).
XRP_TX_PACK_PCT	VT_UI1	Read	Exception transmit pack percentage (see note 4).
<b>When ENABLE_NPS</b>			<b>Node Performance Statistics:</b>
COMM_PCT	VT_UI1	Read	Percentage of this node's communication module processing power currently being used.
AVG_COMM_PCT	VT_UI1	Read	Average percentage of this node's communication module processing power currently being used.
GMI_RX	VT_UI4	Read	Number of GMI messages per second being received by this node.
AVG_GMI_RX	VT_UI4	Read	Average number of GMI messages per second being received by this node.
GMI_TX	VT_UI4	Read	Number of GMI messages per second being transmitted by this node.
AVG_GMI_TX	VT_UI4	Read	Average number of GMI messages per second being transmitted by this node.
XRP_RX	VT_UI4	Read	Number of exception report values per second being received by this node.
AVG_XRP_RX	VT_UI4	Read	Average number of exception report values per second being received by this node.
XRP_TX	VT_UI4	Read	Number of exception report values per second being transmitted by this node.
AVG_XRP_TX	VT_UI4	Read	Average number of exception report values per second being transmitted by this node.
LOOP_TX_BYTE	VT_UI4	Read	Total number of bytes transmitted (forwarded and originated) by this node per second.
LOOP_TX_MSG	VT_UI4	Read	Total number of messages transmitted (forwarded and originated) by this node per second.
AVG_LOOP_TX_MSG	VT_UI4	Read	Average total number of messages transmitted (forwarded and originated) by this node per second.
NODE_RX_BYTE	VT_UI4	Read	Total number of bytes received by (destined for) this node per second.
NODE_RX_MSG	VT_UI4	Read	Total number of messages received by this node per second.
AVG_NODE_RX_MSG	VT_UI4	Read	Average total number of messages received by this node per second.
NODE_TX_BYTE	VT_UI4	Read	Total number of bytes sent (originated) by this node per second.
NODE_TX_MSG	VT_UI4	Read	Total number of messages sent by this node per second.
AVG_NODE_TX_MSG	VT_UI4	Read	Average total number of messages sent by this node per second.
NIS_RX_BYTE	VT_UI4	Read	Total number of bytes transferred on the expander bus from NIS to the communications module per second.

NIS_RX_MSG	VT_UI4	Read	Total number of messages transferred on the expander bus from NIS to the communications module per second.
AVG_NIS_RX_MSG	VT_UI4	Read	Average total number of messages transferred on the expander bus from NIS to the communications module per second.
NIS_TX_BYTE	VT_UI4	Read	Total number of bytes transferred on the expander bus to the NIS from the communications module per second.
NIS_TX_MSG	VT_UI4	Read	Total number of messages transferred on the expander bus to the NIS from the communications module per second.
AVG_NIS_TX_MSG	VT_UI4	Read	Average total number of messages transferred on the expander bus to the NIS from the communications module per second.
MB_RX_BYTE	VT_UI4	Read	Total number of bytes received from the module bus per second (includes bytes of messages received and replies received).
MB_RX_MSG	VT_UI4	Read	Total number of messages received from module bus per second.
AVG_MB_RX_MSG	VT_UI4	Read	Average total number of messages received from module bus per second.
MB_TX_BYTE	VT_UI4	Read	Total number of bytes sent to module bus per second (includes bytes of messages sent and replies sent).
MB_TX_MSG	VT_UI4	Read	Total number of messages sent to module bus per second.
AVG_MB_TX_MSG	VT_UI4	Read	Average total number of messages sent to module bus per second.

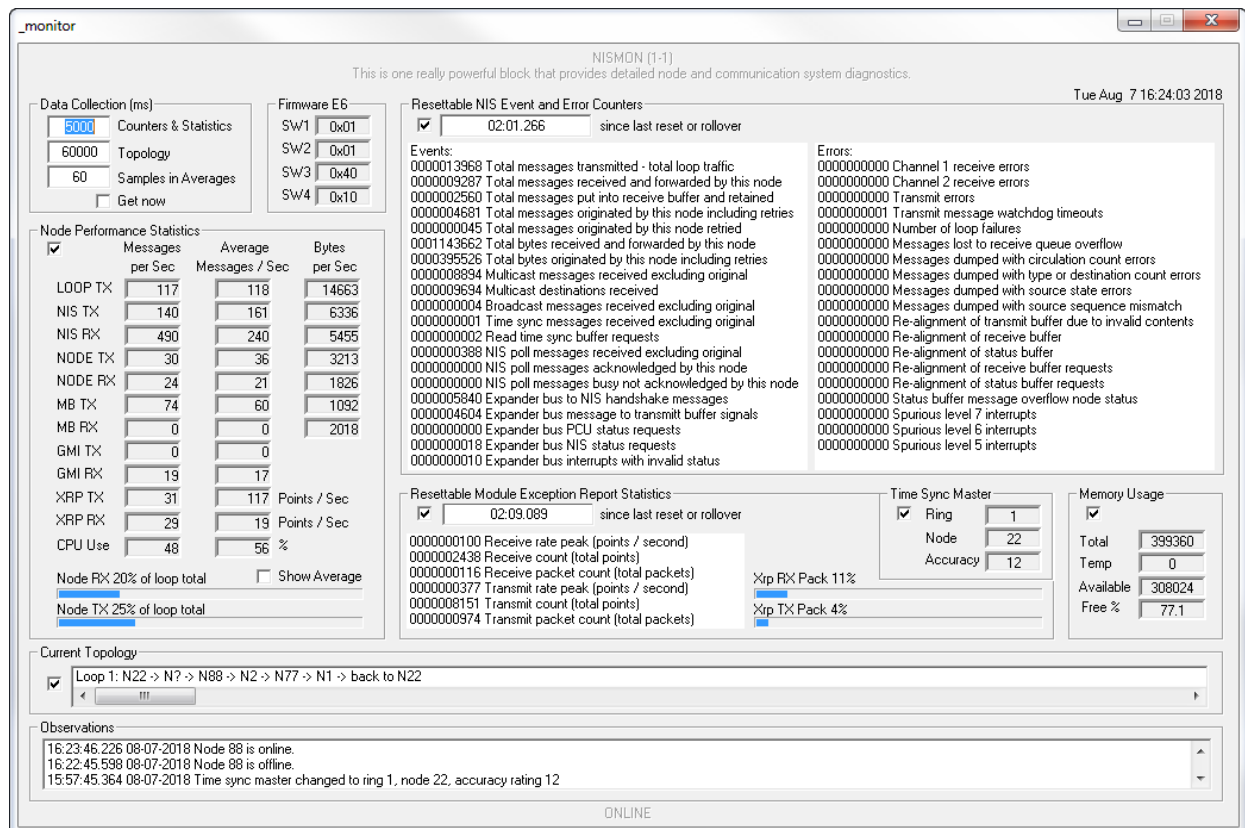
Notes:

- 1.) By writing to these tags, the OPC client can set the NIS node address to be monitored. Use the INTERVAL tag to set how often the communication statistics are read. OPC90 will enforce valid ranges for these tags but it is the client's responsibility to write valid addresses of NIS nodes that actually exist.
- 2.) The INTERVAL tag specifies how often the NIS statistical information are attempted to be read. The minimum value it can be set to is 100 milliseconds. The recommended interval setting is no less than the total number of NISMON and PCUMON blocks times 10000. Setting the INTERVAL to zero, disables reading any information. Up to 73 different statistics can be read based on the settings of the ENABLE\_NIS, ENABLE\_NPS, ENABLE\_MEM, ENABLE\_XR, ENABLE\_SYNC and ENABLE\_MAP tags. Collection of this information is accomplished via poll messages directed at the node. These messages are not as time consuming as poll messages directed to controllers within the PCU node but do consume CIU communication bandwidth and a minimal amount of node bandwidth. Therefore, it is recommended that the INTERVAL tag and ENABLE\_\* tags settings be used judiciously to minimize the impact of this block's data collection on CIU and PCU node communication performance. This is especially important when configuring multiple blocks of this type and using database shadowed OPC90 installations. When specifically diagnosing a particular node, turn on the ENABLE\_\* flags of importance to the diagnosis and reduce the INTERVAL setting to see timely updates of the statistical data for diagnosing performance. After completion of the diagnoses, turn the unneeded ENABLE\_\* flags off or increase the INTERVAL tag setting back to its original setting.
- 3.) All nodes in the system share a node map with each other that defines the topology of the communication loop. Therefore, when the NISMON block is only being used to read the topology and the other data it returns is not required, configuration of only one block per loop is necessary. The reason for this is the topology is shared between all nodes on the loop so regardless of what node it

is requested from it will be the same. The TOPOLOGY tag returns a string defining the topology of the currently addressed loop (ring). The ENABLE\_MAP tag must be set to receive the topology data. Topology data is read every TOPOLOGY\_TIME milliseconds and not tied to the INTERVAL setting. Topology data is read immediately whenever the .GET\_NOW tag is set. The topology data is represented by comma separated values that identify the current loop number, nearest fault node number (set to zero when no faults), a count of total nodes followed by sets of node address and node type code. This list always ends with the node address of the CIU OPC90 is communicating with. The node list is in the message forwarding order. Consider an example string of "1,0,5,30,2,101,0,100,0,2,0,1,0. This string indicates it is for loop 1, no nearest fault node is detected so it is 0, there are 5 nodes in this loop, the first node is 30 and is type 2, the second node is 101 and type 0, the third node is 100, type 0, the four node is 2, type 0 and the last node is 1, type 0. Possible node types are Process control unit (0), Console or computer (2), Bridge (?) and Unknown (?). At the time of this writing the node type codes for Bridge and Unknown are not known. They can be determined in real time when those node types exist in a given system. Note that some node types running old firmware may return zero for its node address. The NISMON faceplate will display that node address as a question mark.

- 4.) The XRP\_RX\_PACK\_PCT and XRP\_TX\_PACK\_PCT tags are calculated points. The exception report count and pack count values are used to compute the number of values per message. Those values are then used to compute what percentage of total possible values per message that represents. A percentage packing rate of 100 would indicate the communication system is running at the busiest it can possibly be. Lower percentages indicate it has plenty of message capacity available. The design of the communication loop is as it gets busy and a message containing exception reports is waiting to get sent the number of values per packet increases. This results in more values per message to be exchanged. When the communication loop is not very busy fewer values per packet are sent since there is communication loop idle space available when it is time for them to get sent. This means smaller packing percentages indicate the loop is not very busy whereas higher factors mean it is.

*Application:* This block's faceplate is very useful for observing the node diagnostic information and determining faults. The faceplate can be viewed by right clicking on the block while in monitor mode and selecting faceplate. It appears as follows:



The “Data Collection” section is used to show and configure how often data is collected. There are two collection periods that can be defined. The first is for the counters and statistics collection and is defined in milliseconds. Setting it to zero will disable collection of those values. The second period is how often the topology is collected and is also defined in milliseconds. Setting it to zero will disable topology collection. The “Get Now” check box instructs the NISMON block to collect the data immediately each time it is clicked. NISMON computes moving averages for the Node Performance Statistics. The moving averages are computed for all tags that show “messages per second” data values. The number of samples (1 – 255) to use in those averages can be defined.

The “Firmware” section shows the firmware revision read from the node’s NIS. It also presents the NIS switch settings as hexadecimal numbers.

The “Resettable NIS Event and Error Counters” section presents various events and error counters read from the node’s NIS. Collection of the data in this section can be disabled by unchecking its check box. This section includes a time since the last NIS hardware reset occurred. This time starts over after 49 days. It can also be reset by writing a zero to it which also resets the event and error counters associated with this section. The purpose of some of the data in this section is obvious but others are hardware specific and have meaning to an ABB field service person. The NISMON observing logic does monitor the error counters and will post information to its log if errors of concern are encountered.

The “Node Performance Statistics” section presents various types of message and byte rates the node is exchanging with the communication loop. It also includes moving averages NISMON is calculating for the messages per second values being collected. These moving averages are useful for trending systems and filter out normal momentary jumps in data rates. The NISMON block uses the LOOP TX, NODE TX and NODE RX message rates to calculate this nodes contribution of messages occurring on the communication loop. The data is presented in terms of percent for messages it is receiving and transmitting. The horizontal progress indicators provide a visual indication of these percentages. The faceplate can display the real time percentages or the averaged data. This is very useful in determining relative levels of node contribution to total communication loop traffic.



The “Resettable Module Exception Report Statistics” section presents information associated with receive and transmit of exception report messages. This data is used to calculate the relative busyness of the communication loop. Collection of the data in this section can be disabled by unchecking its check box. This section includes a time since the last NIS hardware reset occurred. This time starts over after 49 days. It can also be reset by writing a zero to it which also resets the data counters associated with this section. The data in this section reports the peak number of exception reports received and sent per second. It also presents total points and total packets (messages) for received and transmit exception reports. These values are used to compute the percentage packing rate of the receive and transmit messages. The design of the communication loop is as it gets busy and a message containing exception reports is waiting to get sent the number of values per packet increases. This results in more values per message to be exchanged. When the communication loop is not very busy fewer values per packet are sent since there is communication loop idle space available when it is time for them to get sent. This means smaller packing percentages indicate the loop is not very busy whereas higher factors mean it is. The transmit and receive percentage packing factors are shown by vertical progress indicators. They provide a visual indication of how busy the communication loop is. The higher the progress indicator climbs means a busier loop. The faceplate can display the real time or averaged percentage packing factors.

The “Time Sync Master” section shows the ring and node address of the current time sync master along with its accuracy rating. The NISMON observing logic is monitoring the time sync master and will generate a log when it changes.

The “Memory Usage” section presents memory factors associated with the node. Specifically it reports the total memory size, any temporary memory in use, how much memory is available in bytes and percent. The NISMON observing logic is also monitoring percent free memory and will generate a log entry if it becomes low.

The “Current Topology” section shows the order of the current nodes that are online. Collection of this topology information can be disabled by unchecking its check box.

The “Observations” section shows the last thirty observations that have been added to the NISMON log file. This section is useful for seeing the most recent observations but the log file can be viewed using a text editor to see all of them logged each day.

## 7.19 Output Device Driver (ODD)

The ODD OPC90 block interfaces with Bailey consoles as a Bailey Device Driver block (function code 123). Note that this block does not received data from a DD block running in a Bailey controller. Use the OPC90 DD block for that purpose. The ODD is used to indicate on/off control of a discrete device with one or two feedback signals. The operating mode of the device can be reported as auto (controlled by logic) or manual (under operator control).

Restrictions: This OPC block can be utilized with all Bailey interface types except CIU01, serial port module (SPM, CPM02 & CPM03) and computer interface command series (CIC).

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey CIU ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey CIU node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey CIU module address.
ADDR_BLOCK	VT_I2	Config/Read	Block number to establish this point at within the Bailey interface (see note 1).
DD_TYPE	VT_I2	Config/Read	Bailey DD type code to use when establishing this point (see note 2).
MAX_TIME	VT_I4	Config/Read	If OUT never changes, it will be reported at the maximum time interval (seconds) defined by this attribute.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read/Write	Current quality (see note 3) of Bailey values (0-good, 1-bad).
MODE	VT_I2	Read/Write	Mode of this control loop expressed as 0 – Manual, 1 – Auto.
OUT	VT_BOOL	Read/Write	Value of discrete output.
RED_TAG	VT_BOOL	Read/Write	Device red tag indicator.
ALARM	VT_BOOL	Read/Write	Indicates alarm condition.
F1	VT_BOOL	Read/Write	Current state of first feedback signal.
F2	VT_BOOL	Read/Write	Current state of second feedback signal.
FB_STATUS	VT_BOOL	Read/Write	Current state of feedback status 0 – good, 1 - bad.
OVR_STATUS	VT_BOOL	Read/Write	Current state of override status condition.

- 1.) Make sure the block number attribute is unique with respect to the other AOL, DOL, ODD, ORCM, ORMC, ORMSC and OSTN OPC90 blocks associated with the same OPC90 DEVICE block. The block number attribute must also be defined within the range of 1 to maximum number of allowed outputs set up within the associated OPC90 DEVICE block. The Bailey system will receive data from this block at the ring and node address of the Bailey CIU interface, module two and block number defined by the block attribute.
- 2.) The DD type code is a number used by the Bailey consoles to identify characteristics of its DD faceplate display. Currently a value of zero is the only valid code.

- 3.) When the Device block “Set bad quality of max exception timeout” property is enabled, the QUALITY tag of this block will be set bad if writes to the block input(s) do not occur within the Exception Report Output Max Time setting of the block. When this occurs, bad quality will also be written to the CIU and therefore propagated to the users of the block data within the ABB Bailey system. The quality can also be set bad by writing a one (1) to this tag.

*Application:* The OUT tag reports the state of the device as off (0) or on (1). The RED\_TAG tag allows the OPC client to indicate the device is currently under a red tag condition. This block supports mode. When in auto mode, logic controls the state of the OUT tag, therefore, operator commands are ignored. In manual mode the operator writes the requested state of OUT. OPC90 will receive the write request from the Bailey system and transfer this request to the OPC client by writing the new requested state to the OUT tag. When a state change occurs, the OPC client must update the feedback tags (F1 and F2) to reflect successful arrival of the device at the requested state. If feedback cannot be confirmed, the OPC client must set the ALARM tag. Likewise, the OPC client must reset the ALARM flag when a new device state transition is requested and only set it again if feedback cannot be confirmed. The Bailey implementation of the DD block defines the required states of the feedback signals for the device off and on condition and has a feedback wait time. This is the amount of time to wait for feedback confirmation for the commanded state. Therefore, a delay in the setting of the ALARM tag by the OPC client when feedback is not confirmed is acceptable. The OPC client must also set the ALARM tag if good feedback for the current state is lost sometime after it had been confirmed. The FB\_STATUS tag gives an immediate indication as to whether the current state of the feedback signals matches the device state indicated by OUT. The FB\_STATUS tag is similar to the ALARM tag but without the associated feedback waiting time. In other words, the OPC client uses the FB\_STATUS tag as a cycle to cycle indication that the feedback signals match or do not match the device state indicated by OUT. The ALARM tag indicates a mismatch condition only after a feedback waiting time expires after the device transition. The Bailey DD block logic includes a condition in which feedback status is essentially ignored and always considered good regardless of its actual state in comparison to the OUT tag. The OPC client indicates this condition by setting the OVR\_STATUS tag.

## 7.20 Output Multi-State Device Driver (OMSDD)

The OMSDD OPC90 block interfaces with Bailey consoles as a Bailey Multi-State Device Driver block (function code 129). Note that this block does not received data from a MSDD block running in a Bailey controller. Use the OPC90 MSDD block for that purpose. The OMSDD is used to indicate multiple state control of discrete devices having up to four runtime states reported by up to four feedback signals. The operating mode of the device can be reported as auto (controlled by logic) or manual (under operator control).

Restrictions: This OPC block can be utilized with all Bailey interface types except CIU01, serial port module (SPM, CPM02 & CPM03) and computer interface command series (CIC).

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey CIU ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey CIU node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey CIU module address.
ADDR_BLOCK	VT_I2	Config/Read	Block number to establish this point at within the Bailey interface (see note 1).
MSDD_TYPE	VT_I2	Config/Read	Bailey MSDD type code to use when establishing this point (see note 2).
MAX_TIME	VT_I4	Config/Read	If OUT never changes, it will be reported at the maximum time interval (seconds) defined by this attribute.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read/Write	Current quality (see note 3) of Bailey values (0-good, 1-bad).
MODE	VT_I2	Read/Write	Mode of this control loop expressed as 0 – Manual, 1 – Auto.
RED_TAG	VT_BOOL	Read/Write	Device red tag indicator.
GOOD_STATE	VT_I2	Read/Write	Current good state (see note 4).
REQ_STATE	VT_I2	Read/Write	Current requested state (see note 4).
OUT	VT_BOOL	Read/Write	Value of discrete output.
ALARM	VT_BOOL	Read/Write	Indicates alarm condition.
F1	VT_BOOL	Read/Write	Current state of first feedback signal.
F2	VT_BOOL	Read/Write	Current state of second feedback signal.
F3	VT_BOOL	Read/Write	Current state of third feedback signal.
F4	VT_BOOL	Read/Write	Current state of fourth feedback signal.
OVR_CONTROL	VT_BOOL	Read/Write	Current state of control override.
OVR_STATUS	VT_BOOL	Read/Write	Current state of override status condition.

- 1.) ORCM, ORMC, ODD and OMSDD OPC90 blocks associated with the same OPC90 DEVICE block. The block number attribute must also be defined within the range of 1 to maximum number of allowed outputs set up within the associated OPC90 DEVICE block. The Bailey system will receive data from this block at the ring and node address of the Bailey CIU interface, module two and block number defined by the block attribute.

- 2.) The MSDD type code is a number used by the Bailey consoles to identify characteristics of its MSDD faceplate display. Currently a value of zero is the only valid code.
- 3.) When the Device block "Set bad quality of max exception timeout" property is enabled, the QUALITY tag of this block will be set bad if writes to the block input(s) do not occur within the Exception Report Output Max Time setting of the block. When this occurs, bad quality will also be written to the CIU and therefore propagated to the users of the block data within the ABB Bailey system. The quality can also be set bad by writing a one (1) to this tag.
- 4.) The GOOD\_STATE and REQ\_STATE tags are used to indicate the actual (GOOD\_STATE) state of the device and its last requested state (REQ\_STATE). The states are indicated as numbers in the range of 0 through 3. By definition, the zero state is considered the default or safe state of the device. States 1 through 3 are considered the operational states of a device like slow, medium and fast. An operator request to go to a given state is written to the REQ\_STATE tag. Writing the REQ\_STATE tag value to the GOOD\_STATE tag value indicates arriving at the requested state. The two tags not being equal and the ALARM tag not being set indicate a travel condition.

*Application:* The Bailey implementation of the MSDD block contains three discrete outputs used as control signals to a field device typically having more than two states. For example a variable speed motor might have a stopped, slow, medium and fast states. By definition, the MSDD has four device states. These states are the default condition and states one through three. Setup of the MSDD block defines the output values of the three discrete outputs for each of the four device states. For example the default state typically defines the three outputs to be reset, state one could be the first output set and the other two reset and state 3 might be all three outputs are set. For compatibility with the Bailey exception report format this block has an OUT tag, which reports the output value of the first of the three discrete outputs. This output is essentially useless for determining the actual device state but is included for compatibility with existing Bailey implementation. Instead, the Bailey console uses the GOOD\_STATE and REQ\_STATE tags to determine the actual device state and its new requested state. The OPC client writes the actual device state to the GOOD\_STATE tag as a number 0 – 3. A request for a new device state occurs by a write to the REQ\_STATE tag. The RED\_TAG tag allows the OPC client to indicate the device is currently under a red tag condition. This block supports mode. When in auto mode, logic (the OPC client) controls the state change requests written to the REQ\_STATE tag with operator commands to this tag being ignored. In manual mode the operator writes the REQ\_STATE tag. OPC90 will receive the write request from the Bailey system and transfer this request to the OPC client by writing the new requested state to the REQ\_STATE tag. When the OPC client detects a state change has been requested (GOOD\_STATE <> REQ\_STATE), the OPC client must update the feedback tags (F1 through F4) and GOOD\_STATE tag to reflect successful arrival of the device at the requested state. If feedback cannot be confirm, the OPC client must set the ALARM tag and leave the GOOD\_STATE unmodified. Likewise, the OPC client must reset the ALARM flag when a new device state transition is requested and only set it again if feedback cannot be confirmed. The Bailey implementation of the MSDD block defines the required states of the feedback signals for the four possible device states along with a feedback wait time. The feedback wait time is the amount of time to wait before doing feedback confirmation for the requested new state. Therefore, a delay in the setting the ALARM tag, by the OPC client when feedback is not confirmed is acceptable. The OPC client must also set the ALARM tag

if good feedback for the current state is lost sometime after it had been confirmed. The Bailey MSDD block includes a feature that allows a block input to cause a control override condition. This condition overrides operator control, transferring control to logic even when the mode is manual. (Options are available in the MSDD to also transfer the mode to auto when the override control input sets.) The OPC client must indicate the override condition by setting the OVR\_CONTROL tag. The Bailey MSDD block logic includes another condition in which feedback status is essentially ignored and always considered good regardless of its actual state in comparison to the OUT tag. The OPC client indicates this condition by setting the OVR\_STATUS tag.

## 7.21 Output Remote Control Memory (ORCM)

The ORCM OPC90 block interfaces with Bailey consoles as a Remote Control Memory block (function code 62). Note that this block does not received data from a RCM block running in a Bailey controller. Use the OPC90 RCM block for that purpose. The ORCM is used to indicate on/off control of a simple discrete device with a single feedback signal.

Restrictions: This OPC block can be utilized with all Bailey interface types except CIU01, serial port module (SPM, CPM02 & CPM03) and computer interface command series (CIC).

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey CIU ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey CIU node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey CIU module address.
ADDR_BLOCK	VT_I2	Config/Read	Block number to establish this point at within the Bailey interface (see note 1).
ADDR_RING	VT_I2	Config/Read	Bailey CIU ring address.
RCM_TYPE	VT_I2	Config/Read	Bailey RC, type code to use when establishing this point (see note 2).
MAX_TIME	VT_I4	Config/Read	If OUT never changes, it will be reported at the maximum time interval (seconds) defined by this attribute.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read/Write	Current quality (see note 3) of Bailey values (0-good, 1-bad).
OUT	VT_BOOL	Read/Write	Value of discrete output.
RED_TAG	VT_BOOL	Read/Write	Device red tag indicator.
ALARM	VT_BOOL	Read/Write	Indicates alarm condition.
F1	VT_BOOL	Read/Write	Current state of feedback signal.
SET_PERM	VT_BOOL	Read/Write	Set permissive (see note 4).
SET_INPUT	VT_BOOL	Read/Write	Indicates state of set input.
RESET_INPUT	VT_BOOL	Read/Write	Indicates state of reset input.
OVR_STATUS	VT_BOOL	Read/Write	Indicates override status condition.

- 1.) Make sure the block number attribute is unique with respect to the other AOL, DOL, ODD, ORCM, ORMC, ORMSC and OSTN OPC90 blocks associated with the same OPC90 DEVICE block. The block number attribute must also be defined within the range of 1 to maximum number of allowed outputs set up within the associated OPC90 DEVICE block. The Bailey system will receive data from this block at the ring and node address of the Bailey CIU interface, module two and block number defined by the block attribute.
- 2.) The RCM type code is a number used by the Bailey consoles to identify characteristics of its RCM faceplate display. Valid codes can be the sum of: 1 – Output value not displayed, 2 – Feedback signal to be displayed.
- 3.) When the Device block “Set bad quality of max exception timeout” property is enabled, the QUALITY tag of this block will be set bad if writes to the block input(s) do not occur within the Exception Report

Output Max Time setting of the block. When this occurs, bad quality will also be written to the CIU and therefore propagated to the users of the block data within the ABB Bailey system. The quality can also be set bad by writing a one (1) to this tag.

- 4.) SET\_PERM will default to a value of set. The OPC client can disable the ability of Bailey consoles to set the OUT tag by resetting the value of SET\_PERM.

*Application:* The OUT tag reports the state of the device as off (0) or on (1). The RED\_TAG tag allows the OPC client to indicate the device is currently under a red tag condition. This block does not support mode. It is always considered to be in manual mode. Both operators and logic can simultaneously control this block. The operator is allowed to command the OUT tag set when the SET\_PERM tag has been set by the OPC client. OPC90 will receive the write request from the Bailey system and transfer this request to the OPC client by writing the new requested state to the OUT tag. When a state change occurs, the OPC client must update the F1 (feedback) tag to reflect successful arrival of the device at the requested state. F1 must be set when OUT is set and F1 is reset when OUT is reset. The OPC client must set the ALARM tag to indicate the feedback does not agree with the requested device state. Likewise, the OPC client must reset the ALARM flag when a new device state transition is requested and only set it again if feedback cannot be confirmed. The OPC client reports the state of the set and reset logic inputs using the respective SET\_INPUT and RESET\_INPUT tags. The Bailey RCM block includes a feature that allows a block input to control the output value based on the occurrence of a control override condition. This condition occurs when the SET\_PERM, SET\_INPUT and RESET\_INPUT tags are all in the set state. The OPC client indicates the override state condition by setting the OVR\_STATUS tag.



## 7.22 Output Remote Motor Control (ORMC)

The ORMC OPC90 block interfaces with Bailey consoles as a Remote Motor Control block (function code 136). Note that this block does not received data from a RMC block running in a Bailey controller. Use the OPC90 RMC block for that purpose. The ORMC is used to indicate on/off control of a discrete device with interlock signals, up to two reported start permissive signals and up to two reported feedback signals. It also reports bad start conditions due to lack of proper interlock signals and a fault condition upon loss of feedback signals after the device is enabled.

Restrictions: This OPC block can be utilized with all Bailey interface types except CIU01, serial port module (SPM, CPM02 & CPM03) and computer interface command series (CIC).

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey CIU ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey CIU node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey CIU module address.
ADDR_BLOCK	VT_I2	Config/Read	Block number to establish this point at within the Bailey interface (see note 1).
RMC_TYPE	VT_I2	Config/Read	Bailey RMC type code to use when establishing this point (see note 2).
MAX_TIME	VT_I4	Config/Read	If OUT never changes, it will be reported at the maximum time interval (seconds) defined by this attribute.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read/Write	Current quality (see note 3) of Bailey values (0-good, 1-bad).
OUT	VT_BOOL	Read/Write	Value of discrete output.
RED_TAG	VT_BOOL	Read/Write	Device red tag indicator.
ALARM	VT_BOOL	Read/Write	Indicates alarm condition.
F1	VT_BOOL	Read/Write	Current state of first feedback signal.
F2	VT_BOOL	Read/Write	Current state of second feedback signal.
PERM1	VT_BOOL	Read/Write	Current state of first start permissive signal.
PERM2	VT_BOOL	Read/Write	Current state of second start permissive signal.
BAD_START	VT_BOOL	Read/Write	Indicates a bad start condition.
FAULT	VT_BOOL	Read/Write	Indicates a fault has occurred.
ERR_CODE	VT_I2	Read/Write	Indicates error code for bad start and fault conditions (see note 4).
HOLD_STATUS	VT_BOOL	Read/Write	Indicates control status holding state due to an ALARM or FAULT condition.
FAULT_ACK	VT_BOOL	Read/Write	Fault acknowledgment.

- 1.) Make sure the block number attribute is unique with respect to the other AOL, DOL, ODD, ORCM, ORMC, ORMSC and OSTN OPC90 blocks associated with the same OPC90 DEVICE block. The

block number attribute must also be defined within the range of 1 to maximum number of allowed outputs set up within the associated OPC90 DEVICE block. The Bailey system will receive data from this block at the ring and node address of the Bailey CIU interface, module two and block number defined by the block attribute.

- 2.) The RMC type code is a number used by the Bailey consoles to identify characteristics of its RCM faceplate display. Currently a value of zero is the only valid code.
- 3.) When the Device block "Set bad quality of max exception timeout" property is enabled, the QUALITY tag of this block will be set bad if writes to the block input(s) do not occur within the Exception Report Output Max Time setting of the block. When this occurs, bad quality will also be written to the CIU and therefore propagated to the users of the block data within the ABB Bailey system. The quality can also be set bad by writing a one (1) to this tag.
- 4.) The RMC ERR\_CODE codes are defined as: 0 – no error, 1 – stopped by logic, 2 – interlock one is not set, 3 – interlock two is not set, 4 – interlock three is not set, 5 – interlock four is not set, 6 – feedback one is not reset, 7 – feedback two is not reset, 8 – feedback one is set, 9 feedback two is set. The appropriate error code must be written when a bad start or fault condition occurs.

*Application:* The OUT tag reports the state of the device as off (0) or on (1). This block does not support mode. The RED\_TAG tag allows the OPC client to indicate the device is currently under a red tag condition. It is always considered to be in manual mode and under both logic and operator control. Before a device can be commanded to the on state its four interlock inputs and two start permissive inputs must all be set. The interlock inputs are implied to the block operation but are not reported to the Bailey consoles. The start permissive inputs are reported. The OPC client indicates their state using the PERM1 and PERM2 tags. When the device is commanded on without the interlocks or permissive inputs being set, this is considered a bad start. The OPC client indicates a bad start condition using the BAD\_START tag. A bad start can also occur when the feedback signals do not set after the device has been commanded on and a feedback waiting time has expired. The OPC client must also indicate this condition as a bad start using the BAD\_START tag. The OPC client indicates the two feedback signals writing to the F1 and F2 tags. If any interlock or feedback signals are lost after a successful start, the device is considered to be in fault and de-energized. The OPC client indicates this condition by writing a zero to the OUT tag and setting the FAULT tag. When either a bad start or fault condition occurs, the OPC client must use the ERR\_CODE tag to indicate the reason for the condition. Bailey predefines the error codes as indicated in the above note concerning the ERR\_CODE tag. The Bailey implementation of the RMC block defines a hold condition that optionally applies when a bad start arises. The choices are no hold, hold bad start and error indicators until successful start or hold but reset bad start and error indicators on next stop or start request. The OPC client indicates the hold condition by setting the HOLD\_STATUS tag. It should be reset on the next start request. The OPC client uses the FAULT\_ACK tag to indicate a fault has occurred and receive an acknowledgment of that fault when the Bailey console writes a value of zero to FAULT\_ACK.

### 7.23 Output Remote Manual Set Constant (ORMSC)

The ORMSC OPC90 block interfaces with Bailey consoles as a Remote Manual Set Constant block (function code 68). Note that this block does not received data from a RMSC block running in a Bailey controller. Use the OPC90 RMSC block for that purpose. The ORMSC is used to indicate an operator settable analog set point value. Operators can only write set point values to this block within the range specified by its attributes. A set point track indicator indicates logic determination of the set point value.

Restrictions: This OPC block can be utilized with all Bailey interface types except CIU01, serial port module (SPM, CPM02 & CPM03) and computer interface command series (CIC).

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey CIU ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey CIU node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey CIU module address.
ADDR_BLOCK	VT_I2	Config/Read	Block number to establish this point at within the Bailey interface (see note 1).
EU_CODE	VT_I2	Config/Read	Bailey engineering units code to use when establishing this point (see note 2).
ZERO	VT_R4	Config/Read	Zero of SP value in engineering units.
SPAN	VT_R4	Config/Read	Span of SP value in engineering units.
SIG_CHANGE	VT_R4	Config/Read	Amount that SP must change, cycle to cycle before it will be reported to Bailey. Expressed as a percentage of the SPAN.
MAX_TIME	VT_I4	Config/Read	If SP never changes significantly, it will be reported at the maximum time interval (seconds) defined by this attribute.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read/Write	Current quality (see note 3) of Bailey values (0-good, 1-bad).
SP	VT_R4	Read/Write	Set point.
SP_TRACKING	VT_BOOL	Read/Write	Indicates set point tracking when set.

- 1.) Make sure the block number attribute is unique with respect to the other AOL, DOL, ODD, ORCM, ORMC, ORMSC and OSTN OPC90 blocks associated with the same OPC90 DEVICE block. The block number attribute must also be defined within the range of 1 to maximum number of allowed outputs set up within the associated OPC90 DEVICE block. The Bailey system will receive data from this block at the ring and node address of the Bailey CIU interface, module two and block number defined by the block attribute.
- 2.) The engineering units code is a number used by the Bailey consoles to index into a table of text strings representing the engineering units of the set point value.
- 3.) When the Device block "Set bad quality of max exception timeout" property is enabled, the QUALITY tag of this block will be set bad if writes to the block input(s) do not occur within the Exception Report Output Max Time setting of the block. When this occurs, bad quality will also be written to the CIU and

therefore propagated to the users of the block data within the ABB Bailey system. The quality can also be set bad by writing a one (1) to this tag.

*Application:* The OPC client uses the SP tag to report the current block value. The operator is allowed to command the SP to a new value as long as the SP\_TRACKING tag is not set. OPC90 will receive the write request from the Bailey system and transfer it to the OPC client by writing the new requested value to the SP tag. The OPC client is responsible for indicating set point tracking by setting the SP\_TRACKING tag. While in SP\_TRACKING the OPC client also updates the SP tag to the currently tracked value. Updates to the SP tag will be constrained between the configured set point zero and span.

## 7.24 Output Station Control (OSTN)

The OSTN OPC90 block interfaces with Bailey consoles as a Bailey Station block (function codes 21, 22, 23, & 80). Note that this block does not received data from a STN block running in a Bailey controller. Use the OPC90 STN block for that purpose. The OSTN is used to indicate control of an analog PID control loop. The operating mode of the PID control loop can be reported as manual (operator writes the set point and control output values), auto (operator writes the set point value with the control output calculated by the PID algorithm) or cascade (logic determines the set point value with the control output calculated by the PID algorithm).

Restrictions: This OPC block can be utilized with all Bailey interface types except CIU01, serial port module (SPM, CPM02 & CPM03) and computer interface command series (CIC). The other Bailey interfaces do not support being the source of PID tuning parameters. Therefore, PID loop tuning of this point from Bailey consoles is not possible.

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey CIU ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey CIU node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey CIU module address.
ADDR_BLOCK	VT_I2	Config/Read	Block number to establish this point at within the Bailey interface (see note 1).
EU_CODE	VT_I2	Config/Read	Bailey engineering units code to use when establishing this point (see note 2).
STN_TYPE	VT_I2	Config/Read	Bailey station type code to use when establishing this point (see note 3).
ZERO	VT_R4	Config/Read	Zero of PV and SP values in engineering units.
SPAN	VT_R4	Config/Read	Span of PV and SP values in engineering units.
SIG_CHANGE	VT_R4	Config/Read	Amount that PV or SP must change, cycle to cycle before it will be reported to Bailey. Expressed as a percentage of the SPAN.
MAX_TIME	VT_I4	Config/Read	If PV, SP or OUT never changes significantly, it will be reported at the maximum time interval (seconds) defined by this attribute.
HI_LIM	VT_R4	Config/ Read/Write	The setting for the process variable alarm limit used to derive the high alarm condition (see note 4).
LO_LIM	VT_R4	Config/ Read/Write	The setting for the process variable alarm limit used to derive the low alarm condition (see note 4).
DV_LIM	VT_R4	Config/ Read/Write	The setting for the deviation alarm limit between the process variable and set point used to derive the deviation alarm condition (see note 4).
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.

TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read/Write	Current quality (see note 5) of Bailey values (0-good, 1-bad).
HI_ACT	VT_BOOL	Read	PV High alarm active indicator (see note 6).
LO_ACT	VT_BOOL	Read	PV Low alarm active indicator (see note 6).
DV_HI_ACT	VT_BOOL	Read	High deviation alarm active indicator (see note 6).
DV_LO_ACT	VT_BOOL	Read	Low deviation alarm active indicator (see note 6).
MODE	VT_I2	Read/Write	Mode of this control loop expressed as 0 – Manual, 1 – Auto, 2 – Cascade.
MODE_LOCK	VT_BOOL	Read/Write	Indicates locked into current mode when set.
PV	VT_R4	Read/Write	Process variable.
OUT	VT_R4	Read/Write	Control output (range is limited to –5.0 to 105.0).
OUT_TRACKING	VT_BOOL	Read/Write	Indicates output tracking when set.
RED_TAG	VT_BOOL	Read/Write	Device red tag indicator.
SP	VT_R4	Read/Write	Set point.
SP_TRACKING	VT_BOOL	Read/Write	Indicates set point tracking when set.
RI	VT_R4	Read/Write	Ratio index (only used for ratio station types).
AO_BYPASS	VT_BOOL	Read/Write	Indicates analog output bypass when set.
DS_BAD	VT_BOOL	Read/Write	Indicates bad digital control station (Bailey hard DCS station) when set.

- 1.) Make sure the block number attribute is unique with respect to the other AOL, DOL, ODD, ORCM, ORMC, ORMSC and OSTN OPC90 blocks associated with the same OPC90 DEVICE block. The block number attribute must also be defined within the range of 1 to maximum number of allowed outputs set up within the associated OPC90 DEVICE block. The Bailey system will receive data from this block at the ring and node address of the Bailey CIU interface, module two and block number defined by the block attribute.
- 2.) The engineering units code is a number used by the Bailey consoles to index into a table of text strings representing the engineering units of the process variable and set point values.
- 3.) The station type code is a number used by the Bailey consoles to identify characteristics of its station faceplate display. Valid codes are: 1 – Basic with SP, 2 – Ratio, 4 – Cascade, 8 – Basic without SP and 16 – Basic with Bias.
- 4.) This attribute is normally configured when the block is first defined. Since it is associated with the alarm limits of value an OPC client is permitted to write new alarm limits in run time. Care should be taken to not continuously change the limit value since each change necessitates dis-establishing the point from the ABB Bailey interface and than re-establishing it with the new limit value.
- 5.) When the Device block “Set bad quality of max exception timeout” property is enabled, the QUALITY tag of this block will be set bad if writes to the block input(s) do not occur within the Exception Report Output Max Time setting of the block. When this occurs, bad quality will also be written to the CIU and therefore propagated to the users of the block data within the ABB Bailey system. The quality can also be set bad by writing a one (1) to this tag.
- 6.) The alarm active indicators are automatically calculated based on the last process variable and set point value written by the OPC client.

*Application:* Bailey implements analog control loops as a combination of two function blocks. One block is the actual PID algorithm and the other is called a station block (STN) that handles interaction of the PID control values to Bailey consoles. Typical loop variables such as set point, process variable, control output and mode are involved with this interaction. Bailey also makes available a number of auxiliary values associated with the typical values that are also included in the OSTN implementation. The RED\_TAG tag allows the OPC client to indicate the device is currently under a red tag condition. The OPC client reports the mode of the control loop via the MODE tag. Valid modes are manual (0), auto (1) and cascade (2). OPC90 will receive mode write request from the Bailey system and transfer it to the OPC client by writing the new requested mode value to the MODE tag. Bailey logic can lock a control loop into any given mode. The OPC client uses the MODE\_LOCK tag to indicate this condition. The PV tag is used by the OPC client to report the process variable of the control loop. This value is in engineering units and will be clamped between the zero and span defined for the process variable. The OUT tag is used by the OPC client to report the current value of the control output. By definition for Bailey systems, OUT has a range of -5.0 to 105.0 that expresses a percentage. OPC90 will receive control output write request from the Bailey system and when the current mode is manual, transfer the control output to the OPC client by writing it to the OUT tag. Bailey logic can enable control output tracking while in manual mode. The OPC client uses the OUT\_TRACKING tag to indicate this condition. The SP tag is used by the OPC client to report the current value of the set point. This value is in engineering units and will be clamped between the zero and span defined for the set point. OPC90 will receive set point write request from the Bailey system and when the current mode is manual or auto, transfer the set point to the OPC client by writing it to the SP tag. Bailey logic can enable set point tracking while in manual or auto mode. The OPC client uses the SP\_TRACKING tag to indicate this condition. A Bailey STN block can be configured to be a ratio station. Ratio stations are similar to a cascaded loop but differs in its method of set point generation when in cascade mode. The cascaded set point input (wild variable) is multiplied by a ratio adjustment factor (called the ratio index) to determine the actual set point value. The RI tag is used by the OPC client to report the current value of the ratio index. By definition for Bailey systems, the ratio index has a valid range of 0.05 to 10.0. OPC90 will receive ratio index write request from the Bailey system and when the current mode is cascade, transfer the ratio index to the OPC client by writing it to the RI tag. Some older Bailey systems using the station block have also included a "hand held" station called a Digital Control Station that allows plant personnel to take direct control of the loop output. The OPC client uses the AO\_BYPASS tag to indicate this condition has occurred. Likewise, the OPC client uses the DS\_BAD tag to also indicate that the controller has lost communication with the Digital Control Station hand held station.

## 7.25 PCU Communication Statistics Monitor (PCUMON)

The PCUMON OPC90 block is used to monitor detailed communication statistics from a PCU node. These statistics are useful for diagnosing health of a PCU node and to look for overload conditions. A symptom of an overloaded PCU node is a prolonged slowdown in the exception reports being exchanged with that node. This is usually caused by too many GMI (polled type) messages being sent to the PCU node.

Five different overload conditions can be monitored which are:

- communication module processing power utilization becoming to high,
- excessive received GMI messages,
- excessive transmitted GMI messages,
- slowdown of received exception reports,
- slowdown of transmitted exception reports.

When any of these conditions are detected a throttle condition can be flagged. Optionally this block can be setup to cause OPC90 to throttle (slowdown) its GMI (polling) activities with the PCU node being monitored. Doing so could clear the overload condition if OPC90 is generating the excessive amount of GMI messages. Note also there is a Device block parameter called “Max GMI per second per PCU node” that can be configured to automatically limit GMI message generation for all PCU nodes being communicated with. The OPC90 throttle only affects GMI activity. All other OPC90 functionality such as reading and writing exception reports, sending supervisory control commands and exchanging data with OPC clients is not affected. See application section following the block notes for details on using this block’s faceplate and setting up the throttle level settings.

**Cautions:** Only use this block after thoroughly understanding its functionality. Pay attention to its effect on the PCU node and OPC90 when enabling the throttling capability. Be careful with the INTERVAL setting, especially when multiple PCUMON blocks are configured. After completing an analysis of a nodes performance, set INTERVAL to a higher value or zero to disable data collection if it is currently not needed. This block uses polling to get the data which means it does use a portion of the available polling bandwidth between OPC90 and the ABB Bailey CIU device.

**Restrictions:** This OPC90 block is valid for INFI 90, Harmony and Symphony+ systems. The PCU node to be monitored must have INNPM01 firmware revision C\_1 or later. Note that the OPC90 MODSTAT block can be used to determine the firmware revision level of the INNPM01 if it is not known. All INNPM22 firmware revisions are supported for PCU nodes so equipped.

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/ Read/Write	Bailey PCU ring address (see note 1).
ADDR_NODE	VT_I2	Config/ Read/Write	Bailey PCU node address (see note 1).



DEFAULT	VT_BOOL	Read/Write	Setup the default block settings when set.
INTERVAL	VT_UI4	Config/ Read/Write	Desired update interval expressed as milliseconds (see note 2).
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
ENABLE_CSV_LOG	VT_BOOL	Read/Write	Enable logging of collected PCU statistics to a CSV log file (see note 3).
ENABLE_LOOP	VT_BOOL	Read/Write	Enable collection of loop statistics (see note 2).
ENABLE_NODE	VT_BOOL	Read/Write	Enable collection of node statistics (see note 2).
ENABLE_NIS	VT_BOOL	Read/Write	Enable collection of NIS statistics (see note 2).
ENABLE_MB	VT_BOOL	Read/Write	Enable collection of module bus statistics (see note 2).
THROTTLE_MUXCIU	VT_UI4	Read/Write	MUXCIU block throttle factor to be applied when need for OPC90 polling slowdown condition exists. Expressed in milliseconds (see note 4).
THROTTLE_PCUMON	VT_UI4	Read/Write	PCUMON block throttle factor to be applied when need for OPC90 polling slowdown condition exists. Expressed in milliseconds (see note 4).
THROTTLE_POLL	VT_UI4	Read/Write	POLL block throttle factor to be applied when need for OPC90 polling slowdown condition exists. Expressed in milliseconds (see note 4).
THROTTLE_SPEC	VT_UI4	Read/Write	SPEC block throttle factor to be applied when need for OPC90 polling slowdown condition exists. Expressed in milliseconds (see note 4).
THROTTLE_BLK	VT_UI4	Read/Write	BLK block throttle factor to be applied when need for OPC90 polling slowdown condition exists. Expressed in milliseconds (see note 4).
THROTTLE_MODSTAT	VT_BOOL	Read/Write	No throttle when MODSTAT blocks for any controller within the PCU indicate failed or configure mode (see note 5).
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of PCU node communication statistics (0-good, 1-bad).
SAMPLES	VT_UI1	Read/Write	Number of samples to retain for calculating moving averages of the overload conditions being monitored.
THROTTLE_OVR	VT_BOOL	Read	Flags whether or not the throttling of OPC90 polling activities has been overridden because one or more controllers within the PCU are not in execute mode (see note 5).
THROTTLE	VT_BOOL	Read	Flags whether or not the throttling (slowdown) of OPC90 polling activities is in effect due to communication module processing power utilization is too high or exception report rates have dropped too low (see notes 6, 7, 8, 9, 10). Note that throttle declaration does not occur until the device block indicates startup has been completed.
COMM_PCT	VT_I1	Read	Percentage of this node's communication module processing power currently being used.

COMM_PCT_AVG	VT_I1	Read	Moving average of the percentage of this node's communication module processing power currently being used.
COMM_PCT_LEVEL	VT_I1	Read/Write	Communication module processing power utilization percentage level to declare OPC90 polling throttle (see note 6).
GMI_RX	VT_I4	Read	Number of GMI messages per second being received by this node.
GMI_RX_AVG	VT_I4	Read	Moving average of the number of GMI messages per second being received by this node.
GMI_RX_LEVEL	VT_I4	Read/Write	Maximum node received GMI messages per second level to begin OPC90 polling throttle (see note 7).
GMI_TX	VT_I4	Read	Number of GMI messages per second being transmitted by this node.
GMI_TX_AVG	VT_I4	Read	Moving average of the number of GMI messages per second being transmitted by this node.
GMI_TX_LEVEL	VT_I4	Read/Write	Maximum node transmitted GMI messages per second level to begin OPC90 polling throttle (see note 8).
XRP_RX	VT_I4	Read	Number of exception report values per second being received by this node.
XRP_RX_AVG	VT_I4	Read	Moving average of the number of exception report values per second being received by this node.
XRP_RX_LEVEL	VT_I4	Read/Write	Minimum node received exception report values per second level to begin OPC90 polling throttle (see note 9).
XRP_TX	VT_I4	Read	Number of exception report values per second being transmitted by this node.
XRP_TX_AVG	VT_I4	Read	Moving average of the number of exception report values per second being transmitted by this node.
XRP_TX_LEVEL	VT_I4	Read/Write	Minimum node transmitted exception report values per second level to begin OPC90 polling throttle (see note 10).
LOOP_TX_BYTE	VT_I4	Read	Total number of bytes transmitted (forwarded and originated) by this node per second.
LOOP_TX_MSG	VT_I4	Read	Total number of messages transmitted (forwarded and originated) by this node per second.
NODE_RX_BYTE	VT_I4	Read	Total number of bytes received by (destined for) this node per second.
NODE_RX_MSG	VT_I4	Read	Total number of messages received by this node per second.
NODE_TX_BYTE	VT_I4	Read	Total number of bytes sent (originated) by this node per second.
NODE_TX_MSG	VT_I4	Read	Total number of messages sent by this node per second.
NIS_RX_BYTE	VT_I4	Read	Total number of bytes transferred on the expander bus from NIS to the communications module per second.

NIS_RX_MSG	VT_I4	Read	Total number of messages transferred on the expander bus from NIS to the communications module per second.
NIS_TX_BYTE	VT_I4	Read	Total number of bytes transferred on the expander bus to the NIS from the communications module per second.
NIS_TX_MSG	VT_I4	Read	Total number of messages transferred on the expander bus to the NIS from the communications module per second.
MB_RX_BYTE	VT_I4	Read	Total number of bytes received from the module bus per second (includes bytes of messages received and replies received).
MB_RX_MSG	VT_I4	Read	Total number of messages received from module bus per second.
MB_TX_BYTE	VT_I4	Read	Total number of bytes sent to module bus per second (includes bytes of messages sent and replies sent).
MB_TX_MSG	VT_I4	Read	Total number of messages sent to module bus per second.

Notes:

- 5.) By writing to these tags, the OPC client can set the PCU node address to be monitored. Use the INTERVAL tag to set how often the communication statistics are read. OPC90 will enforce valid ranges for these tags but it is the client's responsibility to write valid addresses of PCU nodes that actually exist.
- 6.) The INTERVAL tag specifies how often the node communication statistics are attempted to be read. The minimum value it can be set to is 100 milliseconds. The recommended interval setting is no less than the total number of PCUMON blocks times 10000. Setting the INTERVAL to zero, disables reading of the PCU node communication statistics. Up to 19 different statistics can be read based on the settings of the ENABLE\_LOOP, ENABLE\_NODE, ENABLE\_NIS, ENABLE\_MB and the \*LEVEL tags. Collection of the PCU node communication statistics is accomplished via individual poll messages directed at the node NPM module. These messages are not as time consuming as poll messages directed to controllers within the PCU node but do consume CIU communication bandwidth and a minimal amount of NPM bandwidth. Therefore, it is recommended that the INTERVAL tag, ENABLE\_\* and \*\_LEVEL tags settings be used judiciously to minimize the impact of this block's data collection on CIU and NPM communication performance. This is especially important when configuring multiple blocks of this type and using database shadowed OPC90 installations. When specifically diagnosing a particular node, turn on all of the ENABLE\_\* flags and reduce the INTERVAL setting to see timely updates of the statistical data for diagnosing performance. After completion of the diagnoses, turn the ENABLE\_\* flags off and increase the INTERVAL tag setting back to its original setting.
- 7.) When the ENABLE\_CSV\_LOG tag is enabled, all of the PCU statistics that are being collected are logged to a CSV file called "PCUMON Rx Ny Date Block.CSV" where Rx is the ring number and Ny is the node number of the PCU address, Date is the date of the log file and Block is the name of the PCUMON block generating the log. The location of the CSV file is in the C:\Program Files\OPC90 Server\Log directory. All possible statistics that can be collected are logged to the file. Those statistics that are not currently enabled for collection are logged with a value of negative one (-1). The log file includes a header definition identifying each statistical column. It also contains hourly and daily averages for all logged data followed by the data collected at each interval. The file layout is designed so it can be opened and saved in the Excel format and then statistical analysis of the data can be setup and performed by time which is also a statistical column in the file. Files older than the Device block "Days to Retain Debug Logs" setting will automatically be deleted to avoid using up unnecessary disk space.

- 8.) These throttle factors are applied when any of the conditions have been determined that warrant throttling (slowdown) of OPC90 polling activities. The throttle factors are applied to polled type messages directed to the PCU node being monitored. The THROTTLE\_MUX setting applies to MUXCIU block messages. This factor delays command replies by the specified milliseconds. The THROTTLE\_PCUMON setting applies to this block. This factor is in milliseconds and is added to the blocks INTERVAL setting. The THROTTLE\_POLL setting applies to POLL blocks. This factor is in milliseconds and is added to the POLL block INTERVAL setting. The THROTTLE\_SPEC setting applies to SPEC blocks. This factor is in milliseconds and is a delay added in between each request for specification data. The THROTTLE\_BLK setting applies to BLK blocks. This factor is in milliseconds and is a delay added in between each read or write block request that occurs when module configurations are saved or restored.
- 9.) When this throttle factor is set true all MODSTAT blocks will be searched, looking for those associated with controllers within the PCU node being monitored. If any of those MODSTAT blocks indicate the controller is not in execute mode, declaration of the optimize condition will be suspended. The THROTTLE\_OVR tag indicates the optimization override is in effect. This is important since one or more controllers not in execute mode will not be contributing to exception report traffic which means lower average levels and therefore, a faulty declaration of the throttle state. Obviously for this factor to be effective, MODSTAT blocks for all controllers within the PCU node being monitored should be configured in the OPC90 database.
- 10.) The communication module processing power utilization (COMM\_PCT) is periodically read by this block when COMM\_LEVEL is greater than -1. Each value is submitted to a moving average calculation made available as the COMM\_PCT\_AVG tag. If the value of the COMM\_PCT\_AVG tag exceeds the COMM\_LEVEL tag setting, a THROTTLE condition (slowdown polling) is declared. While in this condition, the throttle factors (see note 3) will be applied to the OPC90 polling activities. If OPC90 polling activities are the cause of the increased communication module processing power utilization, the throttled (slowdown) OPC90 polling activities may lower that utilization. If it doesn't, the COMM\_LEVEL can be adjusted upward to restore normal OPC90 polling rates. Checking for this condition can be disabled by setting COMM\_LEVEL to zero. Setting COMM\_LEVEL to a value of negative one also disables the condition check and causes COMM\_PCT to not get read.
- 11.) The node received GMI message rate (GMI\_RX) is periodically read by this block when GMI\_RX\_LEVEL is greater than -1. Each value is submitted to a moving average calculation made available as the GMI\_RX\_AVG tag. If the value of the GMI\_RX\_AVG tag becomes higher than the GMI\_RX\_LEVEL tag setting, a THROTTLE condition (slowdown polling) is declared. While in this condition, the throttle factors (see note 3) will be applied to the OPC90 polling activities. If OPC90 polling activities are reducing the PCU node's ability to communicate exception reports, the throttled (slowdown) OPC90 polling activities should reduce that influence. If it doesn't, the GMI\_RX\_LEVEL can be adjusted upward to restore normal OPC90 polling rates. Monitoring for this condition can be disabled by setting GMI\_RX\_LEVEL to zero. Setting GMI\_RX\_LEVEL to a value of negative one also disables the condition check and stops periodic reading of GMI\_RX.
- 12.) The node transmitted GMI message rate (GMI\_TX) is periodically read by this block when GMI\_TX\_LEVEL is greater than -1. Each value is submitted to a moving average calculation made available as the GMI\_TX\_AVG tag. If the value of the GMI\_TX\_AVG tag becomes higher than the GMI\_TX\_LEVEL tag setting, a THROTTLE condition (slowdown polling) is declared. While in this condition, the throttle factors (see note 3) will be applied to the OPC90 polling activities. If OPC90 polling activities is reducing the PCU node's ability to communicate exception reports, the throttled (slowdown) OPC90 polling activities should reduce that influence. If it doesn't, the GMI\_TX\_LEVEL can be adjusted upward to restore normal OPC90 polling rates. Monitoring for this condition can be disabled by setting GMI\_TX\_LEVEL to zero. Setting GMI\_TX\_LEVEL to a value of negative one also disables the condition check and stops periodic reading of GMI\_TX.
- 13.) The node received exception report rate (XRP\_RX) is periodically read by this block when XRP\_RX\_LEVEL is greater than -1. Each value is submitted to a moving average calculation made available as the XRP\_RX\_AVG tag. If the value of the XRP\_RX\_AVG tag becomes less than the XRP\_RX\_LEVEL tag setting, a THROTTLE condition (slowdown polling) is declared. While in this

condition, the throttle factors (see note 3) will be applied to the OPC90 polling activities. If OPC90 polling activities is reducing the PCU node's ability to exchange exception reports, the throttled (slowdown) OPC90 polling activities should reduce that influence. If it doesn't, the XRP\_RX\_LEVEL can be adjusted downward to restore normal OPC90 polling rates. Monitoring for this condition can be disabled by setting XRP\_RX\_LEVEL to zero. XRP\_RX\_LEVEL to a value of negative one also disables the condition check and stops periodic reading of XRP\_RX.

- 14.) The node transmitted exception report rate (XRP\_TX) is periodically read by this block when XRP\_TX\_LEVEL is greater than -1. Each value is submitted to a moving average calculation made available as the XRP\_TX\_AVG tag. If the value of the XRP\_TX\_AVG tag becomes less than the XRP\_TX\_LEVEL tag setting a THROTTLE condition (slowdown polling) is declared. While in this condition, the throttle factors (see note 3) will be applied to the OPC90 polling activities. If OPC90 polling activities is reducing the PCU node's ability to exchange exception reports, the throttled (slowdown) OPC90 polling activities should reduce that influence. If it doesn't, the XRP\_TX\_LEVEL can be adjusted downward to restore normal OPC90 polling rates. Monitoring for this condition can be disabled by setting XRP\_TX\_LEVEL to zero. XRP\_TX\_LEVEL to a value of negative one also disables the condition check and stops periodic reading of XRP\_TX.

*Application:* This block's faceplate is very useful for observing the node performance statistics and setting up the various features of operation. The faceplate can be viewed by right clicking on the block while in monitor mode and selecting faceplate. It appears as follows:

The screenshot shows the \_pcumonitor faceplate for PCU 2. It includes sections for Collection settings, Throttle Factors, Watch statistics, and Watch and Throttle levels. The status at the bottom is ONLINE.

**Collection**

Interval (ms): 5000  
Samples: 6

**Throttle Factors (ms)**

BLK: 250  
MUXCIU: 50  
PCUMON: 3000  
POLL: 2000  
SPEC: 1000  
MODSTAT: ☒

**Watch** Tue Aug 7 16:29:03 2018

	Current Bytes / Sec	Current Messages / Sec
<input checked="" type="checkbox"/> LOOP TX	18862	129
<input checked="" type="checkbox"/> NIS TX	17915	193
<input checked="" type="checkbox"/> NIS RX	2322	347
<input checked="" type="checkbox"/> NODE TX	9238	45
<input checked="" type="checkbox"/> NODE RX	123	2
<input checked="" type="checkbox"/> MB TX	461	77
<input checked="" type="checkbox"/> MB RX	3786	0

CSV Log Is On Default Settings

**Watch and Throttle**

	Level	Average	Current	
GMI TX	0	0	0	Messages / Sec
GMI RX	20	1	2	Messages / Sec
XRP TX	65	387	387	Values / Sec
XRP RX	0	0	0	Values / Sec
CPU Use	0	94	94	%

ONLINE

Note that all numeric entries for this dialog are completed using the tab key. Pressing the enter key will dismiss the dialog.

The “Collection” section is used to configure how often the PCU performance statistics are read. Keep in mind that these statistics are individually polled values so it does add to the GMI activity the PCU must handle. To reduce this effect, individual data that is not required can be shutoff and not polled as needed. Setting the interval to zero, completely shuts off all polling activity for the block. The samples setting configures how many samples are used for calculating the averages found in the “Watch and Throttle” section. The range for this setting can be from 1 – 255 samples.

The “Watch” section shows various categories of PCU performance statistics. This data includes current transmit (TX) and receive (RX) data for the various communication paths within the PCU. The data is presented in terms of both bytes per second and messages per second. The communication paths includes total Infinet transmissions (LOOP), network interface slave (NIS), data exchanges specific to the PCU (NODE) and module bus (MB) or Control Way. Collection of data for each of these communication paths can be individually turned off using the checkboxes within this section if that data is not needed or to lessen the GMI impact on the PCU.

The “Throttle Factors” section is used to setup throttle delays for the various block types that generate GMI (polling) activity. So for the above example if a BLK block configuration save is in process and a THROTTLE condition has been detected, the BLK block would stop reading module configuration data as fast possible but instead separate each read with a 500 millisecond delay. Throttle factors can be applied to BLK, MUXCIU, PCUMON, POLL and SPEC blocks. These throttle factors are only applied for those blocks that are directing their GMI (polling) activity to the PCU being monitored by this block and it has been declared to have a THROTTLE condition. The throttle factors are ignored when the MODSTAT throttle override is enabled and MODSTAT blocks are found indicating one or more controllers within the PCU are not in execute mode.

The “CSV Log On” button switches between CSV file logging “On” and “Off” each time it is clicked.

The “Default Settings” button resets all of the settings to default values.

The “Watch and Throttle” section is used to setup throttle conditions to watch for. This data includes transmit (TX) and receive (RX) watches for general polled type messages (GMI) and exception reported (XRP) data. The communication module processing power utilization can also be included as a watch and throttle condition. Watching for any of these conditions can be disabled by entering -1 for the level setting. Setting the level to zero enables watch but not throttle condition detection. Positive level values enable both watch and throttle monitoring. So for example when the PCU receives a large number of GMI RX messages, those requests must be passed on to the various destination controllers within the PCU. That processing takes time and can temporarily slowdown the number of exception reports the PCU reads from each controller, thus slowing down the exception report transmission rate (XRP TX). It is important to note that it is natural for the various watch values to “move” around based on what is happening at any specific moment in time. Therefore the calculated averaged values are used for determining a throttle condition is in effect. Each positive level setting is compared to its corresponding averaged value to determine if a throttle condition should be flagged. The THROTTLE tag is an OR of all of these level condition checks. The faceplate will indicate which watch is contributing to the throttle condition by blinking its label between black and gray. As can be seen for the above example both GMI RX and XRP TX are flashing (notice they are gray in the example). This is because the GMI RX level setting is at 20 messages per second but the average value indicates that level is exceeded so a throttle condition is flagged. This is also true for the XRP TX level setting since it is at 65 but the average value is less which means the PCU transmitted exception report rate has dropped so a throttle condition is declared.

It is important to remember that the throttle factors only apply to the GMI (polling) activity being generated by OPC90. If a node other than OPC90 is causing the throttle condition to occur, slowing down OPC90’s GMI activity will not clear that condition.



## 7.26 Poll Any Block (POLL)

The POLL OPC90 block is used to retrieve polled output values from any Bailey function block output. This includes blocks that have floating point outputs and discrete outputs. Its primary usage is intended for node level data acquisition when the Bailey interface type is a serial port module, CPM02 or CIC. The POLL block can be used to duplicate the Bailey console ad hoc block output queries. Usage of this block with Bailey interfaces other than serial port modules, CPM or CIC should be limited to those few Bailey block output values not currently being exception reported. Data is obtained by polling. Since Bailey is optimized for exception reporting, polling for Bailey outputs is inefficient and should not be used for a large number of values.

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/ Read/Write	Bailey ring address (see note 1).
ADDR_NODE	VT_I2	Config/ Read/Write	Bailey node address (see note 1).
ADDR_MODULE	VT_I2	Config/ Read/Write	Bailey module address (see note 1).
ADDR_BLOCK	VT_I2	Config/ Read/Write	Bailey block address (see note 1).
INTERVAL	VT_I4	Config/ Read/Write	Desired polling interval expressed as milliseconds (see note 1).
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
QUALITY	VT_BOOL	Read	Current quality of Bailey values (0-good, 1-bad).
OUT	VT_R4	Read	The polled Bailey block output value (see note 2).
OUT_TEXT	VT_BSTR	Read	The polled Bailey block output value as a string (see note 2).
ALM	VT_BOOL	Read	High or low alarm active indicator (see note 3).
HI_ACT	VT_BOOL	Read	High alarm active indicator (see note 3).
LO_ACT	VT_BOOL	Read	Low alarm active indicator (see note 3).
HI_DEV_ACT	VT_BOOL	Read	High deviation alarm active indicator (see note 4).
LO_DEV_ACT	VT_BOOL	Read	Low deviation alarm active indicator (see note 4).

### Notes:

- 1.) By writing to these tags, the OPC client can set the address to be polled and how often the value is polled. The server will enforce valid ranges for these tags but it is the client's responsibility to write valid addresses to which block outputs actually exist. Polling is disabled when the OPC client writes zero to the interval tag. Otherwise, the minimum value that can be written is 100 ms.
- 2.) Polling discrete Bailey block outputs will be converted to an equivalent floating point value. If the block output being polled is a string the .OUT will be the count of characters in the string and .OUT\_TEXT will be the actual string.
- 3.) Some polled Bailey floating point block outputs also contain high and low alarm active bits. The alarm level of any such polled block will be reflected in the HI\_ACT and LO\_ACT tags. Some polled Bailey



discrete block outputs also contain an alarm active bit. The alarm level of any such polled block will be reflected in the HI\_ACT tag.

- 4.) Some polled Bailey floating point block outputs also contain high and low deviation alarm active bits. The alarm level of any such polled block will be reflected in the HI\_DEV\_ACT and LO\_DEV\_ACT attributes.

## 7.27 Remote Control Memory (RCM)

The RCM OPC90 block is used to retrieve and control the exception reported output from Bailey Remote Control Memory blocks (function code 62).

Restrictions: This OPC90 block can be utilized with all Bailey interface types except serial port module (SPM & CPM02).

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey module address.
ADDR_BLOCK	VT_I2	Config/Read	Bailey block address.
F1LSD0	VT_BSTR	Config/Read	Feedback 1 state zero logic state descriptor.
F1LSD1	VT_BSTR	Config/Read	Feedback 1 state one logic state descriptor.
OUTLSD0	VT_BSTR	Config/Read	Output state zero logic state descriptor.
OUTLSD1	VT_BSTR	Config/Read	Output state one logic state descriptor.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
FACE_TYPE	VT_I2	Read	Faceplate type (value of RCM block S8).
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey values (0-good, 1-bad).
DISC_ACT	VT_BOOL	Read	Alarm active indicator.
OUT	VT_BOOL	Read/Write	Bailey RCM discrete output value (see note 1).
OUT_TEXT	VT_BSTR	Read/Write	Bailey RCM discrete output value as a string (see note 2).
SUSTAIN	VT_I1	Read/Write	Sustained commands should be used: 0 – No 1 – Yes command OUT with 0 and 1 2 – Yes command OUT with 1 and 2
RESET_INPUT	VT_BOOL	Read	State of reset input.
SET_INPUT	VT_BOOL	Read	State of set input.
RED_CMD	VT_BSTR	Read/Write	Red tag command (see red tag users section).
RED_TAG	VT_BOOL	Read	Red tag indicator (0 – no, 1 – yes).
RED_USERS	VT_BSTR	Read	Red tag users (current user names or codes).
F1	VT_BOOL	Read	State of feedback.
F1_TEXT	VT_BSTR	Read	State of feedback number 1 as a string.
SET_PERM	VT_BOOL	Read	Set permissive.
OVR_STATUS	VT_BOOL	Read	Override status indicator.

### Notes:

- 1.) OPC90 uses pulse reset and pulse set commands when commanding the state of the RCM output. When writing a zero to reset the output, a pulse reset command (5) is actually sent to the RCM block by OPC90. Likewise writing one to set the output, a pulse set command (6) is sent. Some

- older style drivers force the OPC client to sent a 5 and 6 to reset and set the RCM output. For compatibility when converting from such older style drivers to OPC90, it can be configured to also accept values of 5 and 6 for commanding the output reset and set. To enable this ability, the RCM.OUT parameter type must be changed from its default setting of BIT to another multi-bit type such as INT. Right click on the RCM.OUT attribute and select properties to make this change. Note that OPC90 will always report the value of OUT as 0 or 1 regardless of whether it is written as 0, 1, 5 or 6.
- 2.) Text strings can be written to this tag to control the RCM output. Valid text strings are the OUTLSD0 and OUTLSD1 settings along with "0", "1", "5", "6", "Off", "On", "Reset", "Set", "Zero", "One", "False" and "True". Writing any of these strings is case insensitive.

## 7.28 Remote Motor Control (RMC)

The RMC OPC90 block is used to retrieve and control the exception reported output from Bailey Remote Motor Control blocks (function code 136).

Restrictions: This OPC90 block can be utilized with all Bailey interface types except serial port module (SPM & CPM02).

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey module address.
ADDR_BLOCK	VT_I2	Config/Read	Bailey block address.
F1LSD0	VT_BSTR	Config/Read	Feedback 1 state zero logic state descriptor.
F1LSD1	VT_BSTR	Config/Read	Feedback 1 state one logic state descriptor.
F2LSD0	VT_BSTR	Config/Read	Feedback 2 state zero logic state descriptor.
F2LSD1	VT_BSTR	Config/Read	Feedback 2 state one logic state descriptor.
PERM1LSD0	VT_BSTR	Config/Read	Permissive 1 state zero logic state descriptor.
PERM1LSD1	VT_BSTR	Config/Read	Permissive 1 state one logic state descriptor.
PERM2LSD0	VT_BSTR	Config/Read	Permissive 2 state zero logic state descriptor.
PERM2LSD1	VT_BSTR	Config/Read	Permissive 2 state one logic state descriptor.
OUTLSD0	VT_BSTR	Config/Read	Output state zero logic state descriptor.
OUTLSD1	VT_BSTR	Config/Read	Output state one logic state descriptor.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
FACE_TYPE	VT_I2	Read	Faceplate type (value of RMC block S14).
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey values (0-good, 1-bad).
DISC_ACT	VT_BOOL	Read	Alarm active indicator.
OUT	VT_BOOL	Read/Write	The Bailey discrete output value.
OUT_TEXT	VT_BSTR	Read/Write	Bailey RCM discrete output value as a string (see note 1).
RED_CMD	VT_BSTR	Read/Write	Red tag command (see red tag users section).
RED_TAG	VT_BOOL	Read	Red tag indicator (0 – no, 1 – yes).
RED_USERS	VT_BSTR	Read	Red tag users (current user names or codes).
F1	VT_BOOL	Read	State of feedback number 1.
F1_TEXT	VT_BSTR	Read	State of feedback number 1 as a string.
F2	VT_BOOL	Read	State of feedback number 2.
F2_TEXT	VT_BSTR	Read	State of feedback number 2 as a string.
FAULT	VT_BOOL	Read	Fault has occurred indicator.
ERR_CODE	VT_I2	Read	Fault error code (see note 2).
FAULT_ACK	VT_BOOL	Read/Write	Fault acknowledgment (see note 3).
PERM1	VT_BOOL	Read	Permissive #1 indicator.

PERM1_TEXT	VT_BSTR	Read	Permissive #1 indicator as a string.
PERM2	VT_BOOL	Read	Permissive #2 indicator.
PERM2_TEXT	VT_BSTR	Read	Permissive #2 indicator as a string.
BAD_START	VT_BOOL	Read	Bad start indicator.
HOLD_STATUS	VT_BOOL	Read	Status on hold indicator.

Notes:

- 1.) Text strings can be written to this tag to control the RMC output. Valid text strings are the OUTLSD0 and OUTLSD1 settings along with "0", "1", "Off", "On", "Reset", "Set", "Zero", "One", "False" and "True". Writing any of these strings is case insensitive.
- 2.) ERR\_CODE attribute indicates error codes when a bad start or fault condition arises. The following error codes can be returned:

0 - no error	5 - interlock #4 input
1 - stop input	6 - feedback #1 input is 0
2 - interlock #1 input	7 - feedback #2 input is 0
3 - interlock #2 input	8 - feedback #1 input is 1
4 - interlock #3 input	9 - feedback #2 input is 1

- 3.) The FAULT\_ACK tag should be set true (1) to acknowledge a fault or bad start condition and the Bailey RMC block will reset it to false (0) after acknowledgment is accepted.

## 7.29 Remote Manual Set Constant (RMSC)

The RMSC OPC90 block is used to retrieve and control the exception reported output from Bailey Remote Manual Set Constant blocks (function code 68).

Restrictions: This OPC90 block can be utilized with all Bailey interface types except serial port module (SPM & CPM02).

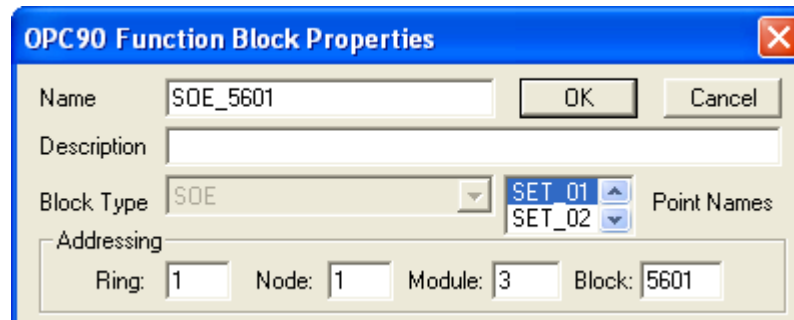
Warning: Do not use a series of these blocks to provide bulk transfer of data between the OPC client and ABB Bailey controller. Instead use the OPC90 AOL block to source the OPC client data and ABB Bailey AI/I (FC 121) or AI/L (FC 26) blocks in the controller to receive the data.

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey module address.
ADDR_BLOCK	VT_I2	Config/Read	Bailey block address.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey values (0-good, 1-bad).
EU	VT_I2	Read	Engineering Units Code.
EU_TEXT	VT_BSTR	Read	Engineering Units String.
OUT	VT_R4	Read/Write	Current output value received from Bailey.
OUT_HI_LIM	VT_R4	Read	OUT high limit.
OUT_LO_LIM	VT_R4	Read	OUT low limit.
SP_TRACKING	VT_BOOL	Read	Set point is tracking indicator.

### 7.30 Sequence of Events (SOE)

The SOE OPC90 block is used to monitor and retrieve sequence of events data from Bailey Sequence of Events Log blocks (function codes 99 and 243). OPC90 supports up to four ABB Bailey sequence of events recorders by allowing definition of four SOE point sets. Each point set consists of 512 points. The Edit->SOE Point Names menu selection is used to define the SOE point sets.

Configuration of this block includes selection of the SOE point name set it is to be associated with.



Restrictions: This OPC90 block can be utilized with all Bailey interface types except serial port module (SPM & CPM02).

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey module address.
ADDR_BLOCK	VT_I2	Config/Read	Bailey block address.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
BLOCK_TYPE	VT_BSTR	Read	Type of SOE log block ('Undefined', 'Standard', 'Summary', 'Pre-fault', 'Post-fault' and 'Snap-shot').
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey sequence of events recorder (0-good, 1-bad). A value of 0 indicates good, otherwise the point address is wrong, the Bailey controller is not communicating with the RA-3800 SER or INSEM01 module.
SOE_DATA	VT_BOOL	Read/Write	Indicates whether or not the SOE block has newly captured SOE data present (0 – empty waiting for a capture event, 1 – capturing data).
OFLOW	VT_BOOL	Read	The SOE data overflowed the Bailey SOE block buffer capacity.

COUNT	VT_I4	Read	Total points logged to the currently selected SOE capture file.
EVENT	VT_BOOL	Read/Write	Signals that a SOE event has been captured (0-armed and waiting, 1-captured and locked).
FILE	VT_BSTR	Read/Write	Indicates / selects the current file name for the captured SOE data.
NEXTFILE	VT_BOOL	Read/Write	Flag used to sequence through the most recent to oldest SOE capture files.
NEXTREC	VT_BOOL	Read/Write	Flag used to sequence through the records of the currently selected SOE capture file..
RECORD	VT_I4	Read/Write	Indicates / selects a record (1 – COUNT) for the currently selected capture file.
POINT_TYPE	VT_BSTR	Read	Point type ('Undefined', 'Standard', 'Summary', 'Pre-fault', 'Post-fault' and 'Snapshot') for current record of selected SOE file.
POINT_NUM	VT_I4	Read	Point number (0 - 511) for current record of selected SOE file.
POINT_NAME	VT_BSTR	Read	Point name for current record of selected SOE file.
POINT_SCAN	VT_BOOL	Read	Point scan state (0 on scan, 1 off scan) for current record of selected SOE file.
POINT_DATE	VT_BSTR	Read	Point capture date month/day/year for current record of selected SOE file.
POINT_TIME	VT_BSTR	Read	Point capture time hour:minute:second.millisecond for current record of selected SOE file.
POINT_Q	VT_BOOL	Read	Point quality (0 good, 1 bad) for current record of selected SOE file.
POINT_ALARM	VT_BOOL	Read	Point alarm state (0 no, 1 yes) for current record of selected SOE file.
POINT_VALUE	VT_BOOL	Read	Point value (0 or 1) for current record of selected SOE file.

*Application:* This block is used to monitor and retrieve sequence of events data from Bailey function code 99 and 243 blocks. Bailey function code 99 blocks can be any block number within the controller block range. On startup, OPC90 reads the Bailey SOE block data to determine its type (FC 99 or 243) and based on the type setup the OPC90 SOE.BLOCK\_TYPE tag. Bailey function code 243 is the INSEM01 executive block. Its base address is always 5000 and is used to retrieve "Standard" sequence of events. Its 5001 block address is used to retrieve "Summary" sequence of events.

The SOE.SOE\_DATA tag indicates the Bailey system is capturing an SOE event when it goes to the one state. When this occurs, OPC90 will read the sequence of event data and store it in SOE files automatically generated within the C:\Program Files\OPC90 Server\SOE directory. The SOE.OFLOW tag indicates the Bailey SOE block experienced an overflow condition (when 1) and the block buffer size must be increased to clear this condition.

SOE.SOE\_DATA will reset when the SOE event capture is complete. The current capture file can be closed and a new one opened by writing a zero to this tag. Note that a value of one can only be written to the SOE.SOE\_DATA tag when the SOE block type is 'Summary' requesting a summary report to be generated. If OPC90 device simulation is enabled, a one and then zero can be written regardless of the SOE block type. This will cause a simulated SOE log file to be generated

By definition file names are the system address of the block (ring, node, module, block separated with an underscore character '\_' ) followed by date of capture followed by a two digit (01 – 99) sequence code. The capture file types are 'STD', 'SUM', 'PRE', 'POS' or 'SNP' which corresponds to standard, summary, pre-



fault, post-fault and snapshot type logs. Note that Bailey blocks based on FC 243 generated by the INSEM01 module only support standard and summary type events.

Consider the following examples:

- 1.) 1\_2\_3\_100\_0420200702.STD
- 2.) 1\_2\_3\_101\_0228200701.SUM
- 3.) 1\_2\_3\_102\_0308200701.PRE
- 4.) 1\_2\_3\_103\_0308200701.POS
- 5.) 1\_2\_3\_104\_1225200699.SNP

Example 1 is the 2nd standard SOE log captured on April 20, 2007 by the SOE block at ring 1, node 2, module 3 and block address 100.

Example 2 is the 1st summary SOE log captured on February 28, 2007 by the SOE block at ring 1, node 2, module 3 and block address 101.

Example 3 is the 1st pre-fault SOE log captured on March 8, 2007 by the SOE block at ring 1, node 2, module 3 and block address 102.

Example 4 is the 1st post-fault SOE log captured on March 8, 2007 by the SOE block at ring 1, node 2, module 3 and block address 103.

Example 5 is the 99th snapshot SOE log captured on December 25<sup>th</sup>, 2006 by the SOE block at ring 1, node 2, module 3 and block address 104.

Note that this naming convention supports up to 99 capture files (a\_a\_a\_a\_00000001.STD – a\_a\_a\_a\_00000099.STD) for each given log type to be recorded per day. After generation of the 99<sup>th</sup> log file (a\_a\_a\_a\_00000099.STD) it will wrap back around to the beginning (a\_a\_a\_a\_00000001.STD). The capture files are stored in the C:\Program Files\OPC90 Server\SOE directory. This directory is also included in the OPC90 database shadowing feature.

The data in the log file is stored in a space separated ASCII format. The first line is a count of the number of SOE records contained by the file. Each subsequent line is a SOE record containing the point number, date string, time string, its quality, alarm state, scan state, value, point type string and point name string. Consider the following example data stored in the file “1\_2\_3\_102\_0201200701.PRE”:

```

4
1 2/01/2007 23:23:50.120 0 0 0 1 Pre-fault Pump_A
2 2/01/2007 23:23:50.125 0 0 1 0 Pre-fault Pump_B
6 2/01/2007 23:23:50.203 0 1 0 1 Pre-fault Compressor_1
9 2/01/2007 23:23:50.987 1 0 0 0 Pre-fault Compressor_2

```

The log is the first one to occur on February 1, 2007 for pre-faulted data points. It contains four SOE records as indicated by the first line.

The first record is point #1 which was logged by the Bailey block on February 1, 2007 at 23:23:50.120. It has good quality, its not in alarm, has not been deleted from the scan and has a value of one. The point type is pre-fault and it called “Pump\_A”.

The second record is point #2 which was logged by the Bailey block on February 1, 2007 at 23:23:50.125. It has good quality, its not in alarm, has been deleted from the scan and has a value of zero. The point type is pre-fault and it called “Pump\_B”.

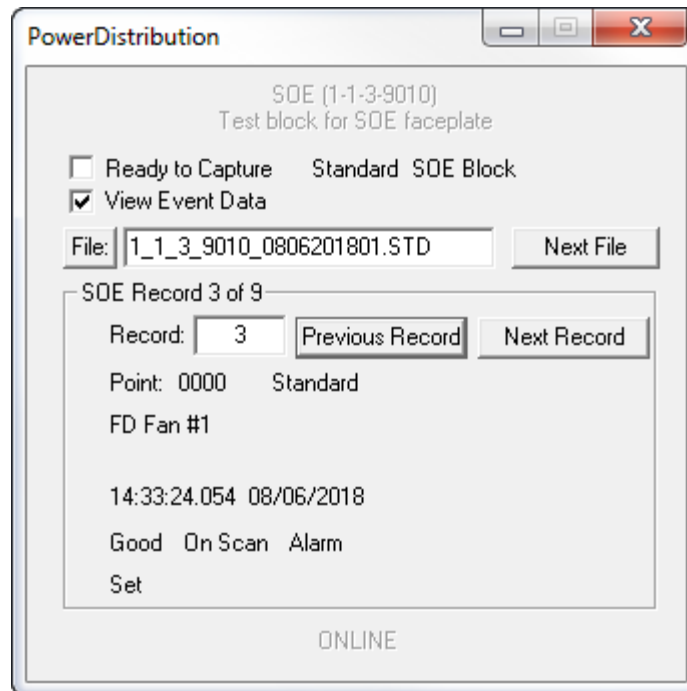
The third record is point #6 which was logged by the Bailey block on February 1, 2007 at 23:23:50.203. It has good quality, its in alarm, has not been deleted from the scan and has a value of one. The point type is pre-fault and it called "Compressor\_1".

The fourth record is point #9 which was logged by the Bailey block on February 1, 2007 at 23:23:50.987. It has bad quality, its not in alarm, has not been deleted from the scan and has a value of zero. The point type is pre-fault and it called "Compressor\_2".

In addition to the above file format, a CSV (comma separated variable) file for each SOE block is generated. The format of this file is compatible for opening by Microsoft Excel. The file name is the system address of the block (ring, node, module, block separated with an underscore character '\_') followed by date of capture followed by a two digit (01 – 99) sequence code with an extension of '.CSV'. The number of points logged is implied by the total lines within the file. Also the first line is the titles for the columns within the file. The CSV capture files are stored in the C:\Program Files\OPC90 Server\SOE directory. This directory is also included in the OPC90 database shadowing feature. Consider the following example CSV file opened with Excel.

	A	B	C	D	E	F	G	H	I
1	POINT	DATE	TIME	QUALITY	ALARM	SCAN	VALUE	TYPE	NAME
2	3	09/26/2007	13:53:43.039	0	0	0	0	Standard	Burner Flame #4
3	5	09/26/2007	13:53:43.040	0	0	0	0	Standard	Undefined
4	9	09/26/2007	13:53:43.041	0	0	0	1	Standard	Undefined
5	11	09/26/2007	13:53:43.042	0	0	0	1	Standard	Undefined
6	15	09/26/2007	13:53:43.043	0	0	0	1	Standard	Undefined
7	19	09/26/2007	13:53:43.044	0	0	0	1	Standard	Undefined
8	29	09/26/2007	13:53:43.045	0	1	0	0	Standard	Undefined
9	33	09/26/2007	13:53:43.046	0	1	0	0	Standard	Undefined
10	116	09/26/2007	13:53:43.047	0	1	0	1	Standard	2EL/XST007

The SOE.EVENT tag is used control browsing of SOE log files. The OPC client sets this tag to indicate its need to look at SOE log file data. When SOE.EVENT is set, the most current SOE log file for the given log type is automatically opened. Note that if the OPC client has not set SOE.EVENT and a Bailey SOE block capture event occurs, the SOE.EVENT tag will automatically set. The SOE.FILE tag indicates the currently opened file, SOE.COUNT the number of records in that file, SOE.RECORD the current record number and SOE\_POINT\_\* provides all of the point information contained by that record. The SOE.NEXTREC tag can be set to advance to the next record within the current file or SOE.RECORD written to a specific record number within the file. The SOE.NEXTFILE tag can be set to advance to the next oldest SOE log file or SOE.FILE can be written to look at a specific file. Note that writes to SOE.FILE do not require path information but can be included if the SOE log files have been moved to an alternative location. Note also that SOE.NEXTREC and SOE.NEXTFILE will automatically reset to indicate the requested action has been completed. All of these tags are utilized to implement the following OPC90 SOE block faceplate.



### 7.31 Specification Block (SPEC)

The SPEC OPC90 block is used to retrieve block specification data for any configured Bailey function block. This block is similar to the OPC90 Server block BLK with the exception that the OPC client does not have to send a command to read the specifications and cannot browse blocks using a NEXT tag. The SPEC block retrieves block specification information through background polling activity. The block address is configured in the RING, NODE, MODULE and BLOCK attributes of this block in the OPC90 Server configuration and thus cannot be changed by OPC Clients. Each time the Bailey block data is polled, the specification values returned are copied to the S01\_VALUE through S63\_VALUE tags. Specification text description data is returned in the corresponding S01 to S63 tags.

Restrictions: None, this OPC90 block can be utilized with all Bailey interface types.

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/ Read/Write	Bailey ring address.
ADDR_NODE	VT_I2	Config/ Read/Write	Bailey node address.
ADDR_MODULE	VT_I2	Config/ Read/Write	Bailey module address.
ADDR_BLOCK	VT_I2	Config/ Read/Write	Bailey block address.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey values (0-good, 1-bad).
ACKNAK	VT_I2	Read	Result of the last block read. A non-zero code indicates an error has occurred.
FC	VT_I2	Read	Function code # returned for the block.
FCNAME	VT_BSTR	Read	Function code name returned for the block.
SX_COUNT	VT_I2	Read	Number of valid specifications for block.
SPEC_FORMAT	VT_I2	Read/Write	Determines how specification data is to be formatted (see note 1).
S01 to S63	VT_BSTR	Read	Text description data for specifications 1 through 63.
S01_STRING to S63_STRING	VT_BSTR	Read/Write	Value for specifications 1 through 63 represented as a string.
S01_TYPE to S63_TYPE	VTI2	Read	Type for specifications 1 through 63 (see note 2).
S01_VALUE to S63_VALUE	VT_R4	Read/Write	Value for specifications 1 through 63.

Since Bailey function blocks can have up to 63 specifications, tags exist for the maximum number of specifications. The actual number of valid specifications for any given block read can be determined by examining SX\_COUNT or the descriptions assigned to them

in tags S01 to S63. Null descriptors indicate the specification number is not valid for the block read.

Some Bailey specifications can be modified while the block is executing. These are called tunable specifications and are indicated as such by the specification description having the 'T' character in front of it. The tags S01\_VALUE through S63\_VALUE can be used to change the value of tunable specifications.

Notes:

- 1.) The format of how the specification data is returned can be selected utilizing the SPEC\_FORMAT attribute. The following choices are available:
  - 0 - Sx Name Value    Format includes spec number, its name and current value
  - 1 - Name Value            Format includes spec name and its current value
  - 2 - Sx Name            Format includes spec number, and its name
  - 3 - Name            Format includes just the name of the spec
- 2.) The specification type is defined according to the following values. Tunable specifications will be indicated by 100 added to these values.
  - 1 = REAL\_2
  - 2 = REAL\_3
  - 3 = BOOLEAN
  - 4 = INTEGER\_1
  - 5 = INTEGER\_2
  - 7 = INTEGER\_4
  - 6 = REAL\_4
  - 8 = STRING

### 7.32 Station PID Control (STN)

The STN OPC90 block is used to retrieve and control the exception reported outputs from Bailey Control Station blocks (function codes 21, 22, 23 & 80).

Restrictions: This OPC90 block can be utilized with all Bailey interface types except serial port module (SPM & CPM02).

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey module address.
ADDR_BLOCK	VT_I2	Config/Read	Bailey block address.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey values: 0=good, 1=bad.
STN_TYPE	VT_I2	Read	Station type: 1=Basic, 2=Ratio, 4=Cascade.
STN_LEVEL	VT_BOOL	Read/Write	Station level (see note 1): 0=Local, 1=Computer.
STN_OK	VT_BOOL	Read/Write	Station computer ok indicator (see note 1): 0=Bad, 1=Ok.
STN_TIMER	VT_I4	Read	Station time in seconds since last computer write (see note 1).
MODE	VT_I2	Read/Write	The mode of the Bailey block (see note 1): 0=Manual 1=Auto 2=Cascade
MODE_LOCK	VT_BOOL	Read	Locked into current mode indicator.
PVM	VT_BOOL	Read/Write	Mode (0=Manual, 1=Auto).
PVMF	VT_BOOL	Read	Mode feedback (0=Manual, 1=Auto).
CASC	VT_BOOL	Read/Write	Mode (0=Manual, 1=Cascade/Ratio).
EU	VT_I2	Read	Engineering Units Code.
EU_TEXT	VT_BSTR	Read	Engineering Units String
PID_BLOCK	VT_I2	Read/Write	PID/ISC block address associated with this control station (see note 2).
PID_BLOCK2	VT_I2	Read/Write	2 <sup>nd</sup> PID/ISC block address associated with this control station (see note 2).
PID_SELECT	VT_I2	Read/Write	PID/ISC selected (0=PID_BLOCK, 1=PID_BLOCK2).
SP_TRACKING	VT_BOOL	Read	Set point tracking indicator.
SP	VT_R4	Read/Write	Current control station set point.

SP_HI_LIM	VT_R4	Read	SP high limit.
SP_LO_LIM	VT_R4	Read	SP low limit.
PV	VT_R4	Read	Current control station process value.
PV_BLOCK	VT_I2	Read	Process value block address.
RI	VT_R4	Read/Write	Current control station ratio index value.
OUT_TRACKING	VT_BOOL	Read	Control output tracking indicator.
OUT	VT_R4	Read/Write	Current control output value received from Bailey.
OUT_HI_LIM	VT_R4	Read/Write	PID/ISC OUT high limit (see note 3).
OUT_LO_LIM	VT_R4	Read/Write	PID/ISC OUT low limit (see note 3).
RED_CMD	VT_BSTR	Read/Write	Red tag command (see red tag users section).
RED_TAG	VT_BOOL	Read	Red tag indicator (0 – no, 1 – yes).
RED_USERS	VT_BSTR	Read	Red tag users (current user names or codes).
HI_ACT	VT_BOOL	Read	PV High alarm active indicator.
LO_ACT	VT_BOOL	Read	PV Low alarm active indicator.
PV_HI_LIM	VT_R4	Read	PV high limit.
PV_LO_LIM	VT_R4	Read	PV low limit.
HI_LIM	VT_R4	Read/Write	The setting for the alarm limit used to detect the PV high alarm condition.
LO_LIM	VT_R4	Read/Write	The setting for the alarm limit used to detect the PV low alarm condition.
DV_LIM	VT_R4	Read/Write	PV Deviation limit.
DV_HI_ACT	VT_BOOL	Read	High deviation alarm active indicator.
DV_LO_ACT	VT_BOOL	Read	Low deviation alarm active indicator.
ALM	VT_BOOL	Read	High or low deviation alarm active indicator.
KFAST	VT_BOOL	Read/Write	Indicates PID/ISC STN block is in fast update mode (see note 5).
KGET	VT_BOOL	Read/Write	Setting this tag requests an immediate read of the PID/ISC tuning parameters (K tags). It is automatically reset when the read has been completed. This tag can't be set until after the STN block PID/ISC function code (.KFC) has been determined.
KFC	VT_I2	Read	PID/ISC function code number.
KTYPE	VT_BSTR	Read	PID/ISC type informational string (interprets KDIR, KIONLY and KTYPE_NUM into string format).
KTYPE_NUM	VT_I2	Read	PID algorithm type Version: 0X = original 1X = new Type: X0 = classical X1 = non-interacting X2 = classical with external reset X3 = manual reset non-interacting
KDIR	VT_BOOL	Read/Write	PID direction switch (see note 4): 0 = reverse acting, SP - PV 1 = direct acting, PV – SP or ISC external reference flag (not writeable for ISC controller): 0 = normal

			1 = use external reference
KIONLY	VT_BOOL	Read/Write	PID set point modifier (see note 4): 0 = normal 1 = integral only on set point change
KI_LIM_TYPE	VT_I2	Read	PID integral limit type 0 = quick saturation recovery 1 = conventional saturation recovery
K	VT_R4	Read/Write	Overall PID/ISC block gain term or ISC predictor process model gain (see note 4).
KP	VT_R4	Read/Write	PID block proportional gain value (see note 4).
KI	VT_R4	Read/Write	PID block integral action time constant or ISC controller tuning time constant (see note 4).
KD	VT_R4	Read/Write	PID block derivative action time constant or ISC process model deadtime (see note 4).
KDLAG	VT_R4	Read/Write	Advance PID (only) block derivative lag action time constant or ISC process model lag time constant (see note 4).
AO_BYPASS	VT_BOOL	Read	Analog output bypass indicator.
DS_BAD	VT_BOOL	Read	Analog output associated with Digital station (hard DCS station) bad indicator (see note 6).

#### Notes:

- 1.) OPC90 can control the STN block in both local and computer level. The STN\_LEVEL tag indicates the current STN block level (0 = local, 1 = computer). The STN block level can be commanded to local or computer by writing the respective values 0 or 1 to STN\_LEVEL. When an OPC client commands the STN block to the computer level it must satisfy the Bailey STN block computer ok timer. This is accomplished by periodically issuing supervisory control writes of the mode, set point or ratio index values. The OPC client can also periodically write a value of 1 to STN\_OK or STN\_LEVEL to satisfy the Bailey STN block computer ok timer (see Device block *AUTOMATIC SEND STN CPU\_OK* setting to delegate management of computer ok to OPC90). The STN\_OK tag reports the status of the Bailey STN block computer ok timer. The OPC client can monitor the STN\_TIMER tag to determine the elapsed time that has transpired since the last computer level supervisory control (mode, set point, ratio index), STN\_OK or STN\_LEVEL write occurred. This timer is active when STN\_LEVEL indicates computer level and STN\_OK indicates ok. When writing mode, OPC90 first checks the current level of the STN block and sends out the appropriate new mode command based on the current level. So for example if you command the STN block mode to auto by writing a value of 1 and the STN level is currently computer manual, OPC90 will send out the appropriate mode command for computer auto. The important thing to remember is the values used to indicate and control Bailey STN block mode are always 0 = manual, 1 = auto, 2 = cascade regardless of STN block level (local or computer).
- 2.) The default value of zero instructs the driver to automatically determine the PID/ISC block number associated with the Bailey control station block. It follows the block input stream starting with the STN S3 input looking for the existence of a PID of type FC 18, 19, 156 or ISC (FC 160). If one is found, it's block number is written to this tag. Otherwise, the value is set to -1 to indicate PID/ISC block not found. This will be the case if a PID/ISC is not configured in the STN input stream. If a 2<sup>nd</sup> PID/ISC block is found associated with the STN, it's number is written to the PID\_BLOCK2 tag. If neither is found, the user can write the appropriate block number to these tags (PID\_BLOCK, PID\_BLOCK2) to make the other STN.K\* tuning items functional. It is important to note that if the other STN.K\* tags are not functioning, check the value of the PID\_BLOCK tag, making sure it has a valid PID/ISC block number. Until an actual PID/ISC block is determined, the OUT\_HI\_LIM, OUT\_LO\_LIM, K, KFC, KTYPE, KDIR, KIONLY, KP, KI, KD and KDLAG remain unknown. In the case where two PID/ISC blocks are determined, the STN.K\* tags will indicate the values associated with PID\_BLOCK when the

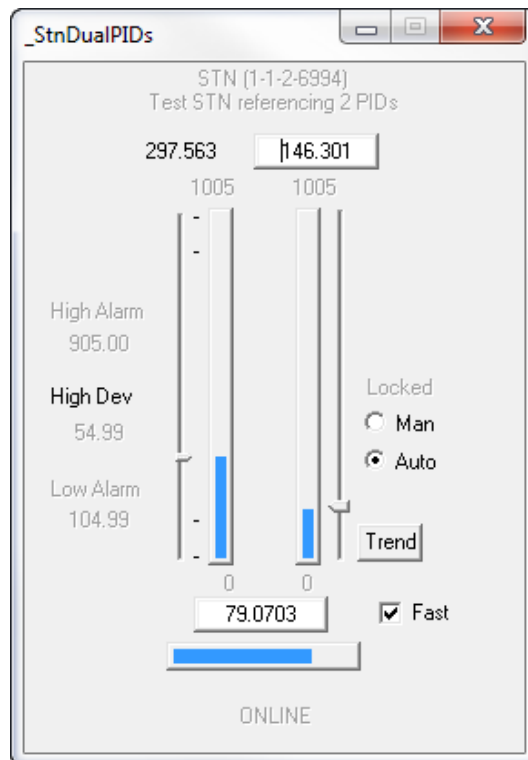


PID\_SELECT tag is 0. When the PID\_SELECT tag is 1 the STN.K\* tags will indicate the values associated with the PID\_BLOCK2.

- 3.) OUT\_HI\_LIM and OUT\_LO\_LIM are the high and low limit specifications read from the Bailey PID/ISC block associated with the Bailey Control Station block. Writing these values from OPC90 causes the driver to automatically tune them in the appropriate Bailey PID/ISC block specification.
- 4.) K, KDIR, KIONLY, KP, KI, KD and KDLAG are specifications read from the Bailey PID/ISC block associated with the Bailey Control Station block. Writing these values from OPC90 causes the driver to automatically tune them in the appropriate Bailey PID/ISC block specification.
- 5.) When any of the K, KDIR, KIONLY, KP, KI, KD and KDLAG tags are written (even writing the same value), OPC90 Server automatically puts the STN block into the “fast update” state. This state causes OPC90 Server to acquire the STN.PV and STN.CO values with a polling command instead of exception reporting. This operation is very useful for loop tuning software. The duration for the fast update state and polling interval are device settings (see DEVICE block). The KFAST attribute indicates the fast update state is in effect. The fast update interval can be refreshed by writing a one to the KFAST attribute or writing a new value to any of the other K tags (see note 4). The fast update interval is canceled automatically when the fast update interval for the STN block has transpired or a value of zero is written to the KFAST attribute. Some applications may want to have direct control of the KFAST feature and not have writes to the other K\* values automatically turn it on. This can be accomplished by turning off the DEVICE block “Set .KFAST on .K\* writes” property.
- 6.) When the analog output associated with a digital control station is bad, the STN block is locked into manual mode. Attempted mode changes when the DS\_BAD value indicates bad will be rejected.

### 7.32.1 Station PID Control (STN) Faceplate

The STN OPC90 block faceplate allows monitoring and supervisory control of the PID loop.



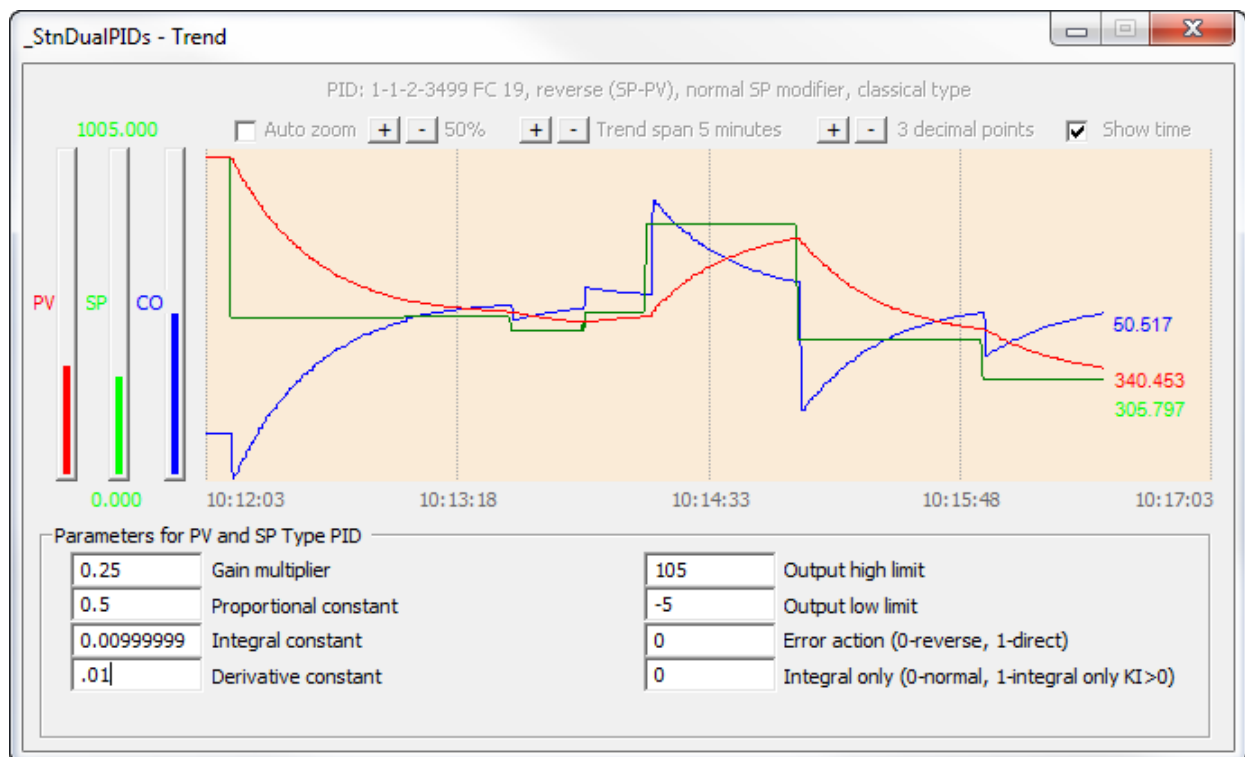
In addition to monitoring the PID loop, this faceplate can display a trend of the control variables by clicking on the “Trend” button. If the trend button isn’t visible that indicates the STN block wasn’t able to

determine the PID block address. A new search can be initiated by writing a 0 to the STN.PID\_BLOCK parameter. If the new search finds the PID, the trend button will be displayed. If the PID block address is known, it can alternatively be written to the STN.PID\_BLOCK parameter to bypass the search. The PID\_BLOCK value is remembered when the OPC90 database is saved.

Notice the “Fast” checkbox. Setting this option instructs the STN block to bypass exception report reading of the process variable and control output. Instead, those values are polled. The poll rate and how long fast update remains enabled are parameter settings in the DEVICE block.

### 7.32.2 Station PID Control (STN) Trend and Tune Faceplate

Clicking on the STN OPC90 block faceplate “Trend” button displays the following faceplate. It is handy for monitoring the PID control action and doing manual tuning of the loop.



The background color of the trend is changed by moving the cursor anywhere in the trend area and click the left mouse button. The trend controls along the top of the trend area is shown or hidden by moving the cursor anywhere in the trend area and click the right mouse button.

The trend controls allow the following items to be modified:

- show / hide trend time display,
- how many decimal points (0-3) to show with the trended values,
- adjust the trend span (1-60 minutes),
- enable auto zoom and adjust its percentage (0-99).

The auto zoom feature is useful when watching very small differences between set point and process variable. It centers the set point value along the y axis by auto scaling the range of set point. How much scaling is defined by a percentage.

The PID tuning parameters can be manually adjusted by writing a new value to any of the indicated parameters. OPC90 automatically tunes the new value to the corresponding parameter in the running

PID loop. The effect of the tune can be observed when a small set point change is made and watching the trended control variables. If the change results in undesirable action, the values can be re-adjusted manually or simply dismiss the faceplate and it will ask the user whether or not to keep the tuning changes.

### **TXT Text Selector (TXT)**

The TXT OPC90 block is used to retrieve the exception reported output from Bailey Text Selector blocks (function code 151).

Restrictions: This OPC90 block can be utilized with all Bailey interface types except serial port module (SPM & CPM02).

<b>TAGNAME</b>	<b>TYPE</b>	<b>ACCESS</b>	<b>DESCRIPTION</b>
ADDR_RING	VT_I2	Config/Read	Bailey ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey module address.
ADDR_BLOCK	VT_I2	Config/Read	Bailey block address.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey values (0-good, 1-bad).
MSG_OUT	VT_I4	Read	Message number to be displayed.
MSG_OUT_TEXT	VT_BSTR	Read	Message number to be displayed as a string.
COLOR	VT_I2	Read	Color selection for the message.
BLINK	VT_BOOL	Read	Blink the message flag.

### 7.33 Text String (TEXTSTR)

The TEXTSTR OPC90 block is used to retrieve the exception reported output from Bailey Text String blocks (function code 194).

Restrictions: This OPC block is only valid for Bailey interface types INICI03, INICI12 and INICI13.

TAGNAME	TYPE	ACCESS	DESCRIPTION
ADDR_RING	VT_I2	Config/Read	Bailey ring address.
ADDR_NODE	VT_I2	Config/Read	Bailey node address.
ADDR_MODULE	VT_I2	Config/Read	Bailey module address.
ADDR_BLOCK	VT_I2	Config/Read	Bailey block address.
MY_BLOCK_TYPE	VT_BSTR	Read	Indicates the OPC90 block type.
MESSAGE	VT_BSTR	Read	Indicates the block operational message.
TAG	VT_BSTR	Read	Indicates the block name.
TAGLONG	VT_BSTR	Read	Indicates the block long OPC path name.
TAGDESC	VT_BSTR	Read	Indicates the block descriptor.
OPC_STATUS	VT_I2	Read	OPC status value (see OPC status section)
QUALITY	VT_BOOL	Read	Current quality of Bailey values (0-good, 1-bad).
QUAL_OVR	VT_BOOL	Read	Quality override (0-Off, 1-On).
EU	VT_I2	Read	Engineering Units Code.
EU_TEXT	VT_BSTR	Read	Engineering Units String.
MODE	VT_I2	Read/Write	The mode of the Bailey block. 0 - Manual 1 - Auto
MODE_LOCK	VT_BOOL	Read	Mode lock (0-Off, 1-On).
ALARM	VT_BOOL	Read	Alarm active indicator.
ALARM_REQ	VT_BOOL	Read/Write	Alarm request (0-No, 1-Yes) (see note 1).
TEXTSTR	VT_BSTR	Read/Write	Text string value (see note 2).
TEXTSTR_LOCK	VT_BOOL	Read	Text string lock (0-Off, 1-On).
TEXTSTR_MAX	VT_I2	Read	Maximum text string size allowed.
TEXTSTR_SEQ	VT_I2	Read	Comm system text string sequence number.

Notes:

- 3.) When ALARM\_REQ is set (1) and a write occurs to TEXTSTR, the write will specify the string is in alarm.
- 4.) The text string value can be written when the block is in manual mode, the text string lock is not on and the TEXTSTR application and / or block configuration supports writes.

## 8 OPC Status

All blocks that exchange data with the ABB Bailey system include an OPC tag called OPC\_STATUS. This tag provides the value of the OPC status. The value is defined with three sub fields. The two most significant bits define the OPC quality, The next four bits define the sub-status for the current quality. The two least significant bits define the limit bits for the current quality and sub-status. The tag value is defined as the logical OR of the following definitions:

```
// OPC Quality codes
#define QUAL_BAD 0x0000
#define QUAL_UNCERTAIN 0x0040
#define QUAL_GOOD 0x00C0

// Limit Bit Field
#define QUAL_NOT_LIMITED 0x0000
#define QUAL_LOW_LIMITED 0x0001
#define QUAL_HIGH_LIMITED 0x0002
#define QUAL_CONSTANT 0x0003

// Substatus code for all
#define QUAL_NON_SPECIFIC 0x0000

// Substatus code for GOOD
#define QUAL_LOCAL_OVERRIDE 0x0018

// Substatus code for BAD
#define QUAL_CONFIGURATION_ERROR 0x0004
#define QUAL_NOT_CONNECTED 0x0008
#define QUAL_DEVICE_FAILURE 0x000C
#define QUAL_SENSOR_FAILURE 0x0010
#define QUAL_LAST_KNOWN_VALUE 0x0014
#define QUAL_COMM_FAILURE 0x0018
#define QUAL_OUT_OF_SERVICE 0x001C

// Substatus codes for UNCERTAIN
#define QUAL_LAST_USABLE_VALUE 0x0004
#define QUAL_SENSOR_NOT_ACCURATE 0x0010
#define QUAL_ENG_UNITS_EXCEEDED 0x0014
#define QUAL_SUB_NORMAL 0x0018
```

So the OPC status value of bad, out of service would equal decimal 28 (hex 1C) which is QUAL\_BAD | with QUAL\_OUT\_OF\_SERVICE.

## 9 CSV File Formats

This section is provided to help the user quickly setup an OPC90 Server configuration. A CSV file is an ASCII Text based file containing fields delimited by a comma. Microsoft Excel can be used to create this file and simply export it in the CSV file format. Each line (row in Excel) in the file declares exactly one OPC90 block. OPC90 groups and tags are not defined within the file. Groups and tags will be created automatically based on the block name and type. File formats for each line in the CSV file are given below. If you are still unsure about the exact format after reviewing the following tables, consult the SamplePoints.CSV included with the OPC90 installation. You can also configure an example of a block using OPC90 Server itself and then export that database to a CSV file.

In addition to configuring OPC90 blocks, the CSV file supports the definition of the engineering units map and text message map. These maps are normally defined using the OPC90 Edit->Engineering Units and Edit->Text Messages dialogs. Definitions within the CSV file allow bulk definition of these two maps.

The format for defining the maps and block definitions are presented by the following tables.

Map Definition	Fields
EUMAP	EU Text, EUMAP, EU Code (valid range is 0 – 255)
REDUSER	Name, User number (1-128), User Code (1 – 65535)
TEXTMAP	Text Message, TEXTMAP, Text Message Number (valid range is 0 – 10000)
SOEPOINT	Point Name, SOEPOINT, Point Number (0-511), Point Set (1-4)

Block	Fields
AIL	Block Name, Block Type, Location, Description, Read Only Flag, Alarm hold time, Alarm hold percent deadband
AOL	Block Name, Block Type, Location (only block #), Description, Max Time, Initial Value, EU code, Zero, Span, Significant Change, High Alarm Limit, Low Alarm Limit
BLK	Block Name, Block Type, Location, Description, Password
DAANG	Block Name, Block Type, Location, Description, Read Only Flag
DADIG	Block Name, Block Type, Location, Description, OUTLSD0, OUTLSD1, Read Only Flag
DD	Block Name, Block Type, Location, Description, F1LSD0, F1LSD1, F2LSD0, F2LSD1, OUTLSD0, OUTLSD1, Read Only Flag
DEVICE	Device Name, Block Type, Port, Description, Max Outputs, Watchdog Timer, Import Exception Report Update Rate, Max Block Specification Reads per Second, STN Fast Update Period, STN Fast Update period, Exception Report Screen Flag, Establish Points Online Flag, System Type Flag (0 = Infi90, 1 = Network 90), Get Set Time Flag, Get Time Flag, Time Warp Flag, Use DCS Time Stamp Flag, Lock Flag, Enable Simulation Flag, Simulate Device Input Blocks Flag, Simulation AOL and DOL Block Inputs, Simulate Factor, Tag Startup Status (1 = Good, 2 = Uncertain, 4 = Bad), Continuous OPC client Update Flag, Exception Report Write Confirmation flag, Group Output Writes flag, Output Exception Report Update Rate, Log Save Days, Enhanced Analog Precision Flag, MSDD Pulse Out Handling Flag, Override DCS Time Stamp Flag, SOE Save Days, Automatic STN CPU_OK Flag, Extended STN.MODE range, Tag Startup Complete on XRP, Set STN.KFAST on STN.K* Writes, Set Output Block Types QUALITY Bad When No Inputs, Max GMI per Second, Max Keys to Use, Turbo Poll Enable, Accuracy Rating, Set Last Tag Value, Auto Disconnect, Block Specification Read Period (seconds), Allow Bad Quality Turbo Polling

DIL	Block Name, Block Type, Location, Description, OUTLSD0, OUTLSD1, Alarm hold time
DOL	Block Name, Block Type, Location (only block #), Description, Max Time, Initial Value, Alarm State
HAI	Block Name, Block Type, Location, Description, Read Only Flag
HAO	Block Name, Block Type, Location, Description, Read Only Flag
HDI	Block Name, Block Type, Location, Description, OUTLSD0, OUTLSD1
HDO	Block Name, Block Type, Location, Description, OUTLSD0, OUTLSD1
MODSTAT	Block Name, Block Type, Location (ring, node, module), Description
MSDD	Block Name, Block Type, Location, Description, F1LSD0, F1LSD1, F2LSD0, F2LSD1, F3LSD0, F3LSD1, F4LSD0, F4LSD1, GSLSD0, GSLSD1, GSLSD2, GSLS3, Read Only Flag
MUXCIU	Block Name, Block Type, Port, Description, Read Only, Allow R4, Enable Parallel Pre-fetch
NISMON	Block Name, Block Type, Location, Description, Update Interval, Topology Interval, Samples, EnableNis, EnableNps, EnableMem, EnableXrp, EnableMap, EnableSync
ODD	Block Name, Block Type, Location (only block #), Description, Max Time, 0, Faceplate Type
OMSDD	Block Name, Block Type, Location (only block #), Description, Max Time, 0, Faceplate Type
ORCM	Block Name, Block Type, Location (only block #), Description, Max Time, 0, Faceplate Type
ORMC	Block Name, Block Type, Location (only block #), Description, Max Time, 0, Faceplate Type
ORMSC	Block Name, Block Type, Location (only block #), Description, Max Time, EU code, Zero, Span, Significant Change
OSTN	Block Name, Block Type, Location (only block #), Description, Max Time, EU code, Zero, Span, Significant Change, High Alarm Limit, Low Alarm Limit, Deviation Alarm Limit, Station Type
PCUMON	Block Name, Block Type, Location, Description, Update Interval, Samples, EnableLoop, EnableNode, EnableNis, EnableMb, CommLevel, GmiRxLevel, GmiTxLevel, XrpRxLevel, XrpTxLevel, FactorMuxCiu, FactorPoll, FactorSpec, FactorBlk, FactorPcuMon, FactorModStat
POLL	Block Name, Block Type, Location, Description, Poll Interval
RCM	Block Name, Block Type, Location, Description, F1LSD0, F1LSD1, OUTLSD0, OUTLSD1, Read Only Flag
RMC	Block Name, Block Type, Location, Description, F1LSD0, F1LSD1, F2LSD0, F2LSD1, PERM1LSD0, PERM1LSD1, PERM2LSD0, PERM2LSD1, OUTLSD0, OUTLSD1, Read Only Flag
RMSC	Block Name, Block Type, Location, Description, Read Only Flag
SPEC	Block Name, Block Type, Location, Description, Read Only Flag
STN	Block Name, Block Type, Location, Description, Read Only Flag
TEXT	Block Name, Block Type, Location, Description
TEXTSTR	Block Name, Block Type, Location, Description, Read Only Flag

The following table is a guide to understanding the various block fields used when defining OPC90 blocks

FIELD NAME	FIELD DESCRIPTION	Examples
Block Name	Contains the case-insensitive full block name (not the tagname). The full block name includes the device name followed by all groups followed lastly by the block name. (see notes)	CIU4.Group1.STN1 Device1.DIL23 CIU4 CIU4.Unit1.Boiler.Turbine.Mtr1
Block Type	Specifies the case-insensitive OPC90 Server Block Type as one of the following strings: AIL, AOL, BLK, DEVICE, DAANG, DADIG, DIL, DOL, HAI, HAO, HDI, HDO, MODSTAT, MSDD, ODD, OMSDD, ORCM,	AIL, Ail, ail, stn, STN, AOL, RMSC



	ORMC, ORMSC, OSTN, POLL, RCM, RMC, RMSC, SPEC, STN, TXT, TEXTSTR	
Location	Specifies the Bailey address as: Bailey ring address. Bailey node address. Bailey ring address. Bailey block address. Address fields must be separated by a dot not a comma. For AOL, DOL, ODD, OMSDD, ORCM, ORMC, ORMSC and OSTN blocks simply require the bailey block #. For MODSTAT blocks simply require the ring.node.module The address can be encased in quotation marks if desired.	1.1.2.1000 "1.1.2.1000" 1.1.4.200 200 (for AOL, DOL, ODD, OMSDD, ORCM, ORMC, ORMSC, OSTN) "200" (for AOL, DOL, ODD, OMSDD, ORCM, ORMC, ORMSC, OSTN) 1.1.2.200 (for AOL, DOL, ODD, OMSDD, ORCM, ORMC, ORMSC, OSTN where the 1.1.2 is disregarded) COM1: (for DEVICE) 1.1.5 (for MODSTAT) 1.1.5.12 (for MODSTAT where block 12 is disregarded)
Port Description	For DEVICE blocks this field specifies the Com Port such as COM1:, COM2:, COM3: or COM4:.	
Description	User-Defined description of the block. This field cannot have any commas in it unless it is encased in double quotes. For example <i>"This, is a block description with a comma in it"</i> . Note that the double quotes will be parsed out. Don't use any commas in the description if it is not encased by double quotes.	Unit 1 Boiler Exhaust Flow PID1024 Test Block Description "Unit 1,2 Combined MW"
MaxTime or Interval	Only used for AOL, DOL, ODD, OMSDD, ORCM, ORMC, ORMSC, OSTN and POLL blocks. Specifies the MaxTime (in seconds) for AOL, DOL, ODD, OMSDD, ORCM, ORMC, ORMSC and OSTN blocks.	10
Poll Interval	Used for POLL blocks specifying the polling interval in milliseconds.	1000
Read only flag	Used for AIL, DAANG, DADIG, DD, HAI, HAO, MSDD, MUXCIU, RCM, RMC, RMSC, SPEC, STN and TEXTSTR blocks to set its operation to read only. When this flag is set, all writes to the block that change data in ABB Bailey controllers are rejected.	0 = no or 1 = read only
InitialValue	Only for AOL and DOL blocks. Specifies the initial value for these blocks. Must be defined as 0 for ODD, OMSDD, ORCM, ORMC block. Must not be defined for ORMSC and OSTN blocks.	12.456 0 or 1 (for DOL)
EU or AlarmState or FaceplateType	Specifies the EU Units integer value for AOL, ORMSC and OSTN blocks. For DOL blocks this field specifies the AlarmState: 0-alarm when 0 1-alarm when 1 2-alarming disabled.	10 0,1 or 2 (for DOL)

	For ODD, OMSDD, ORCM and ORMC blocks this field specifies the faceplate type code.	
Zero	For AOL, ORMSC and OSTN only. Specifies the value of the zero in engineering units.	-1000.0
Span	For AOL, ORMSC and OSTN only. Specifies the value of the span in engineering units.	2000.0
Significant Change	For AOL, ORMSC and OSTN only. Specifies the value of the significant change in engineering units.	1.5
HI LIMIT	For AOL and OSTN only. Specifies the value for the alarm limit used to derive the high alarm condition.	900.0
LO LIMIT	For AOL and OSTN only. Specifies the value for the alarm limit used to derive the low alarm condition.	100.0
DV ALARM	For OSTN only. Specifies the Station deviation alarm (difference between SP and PV).	10.0
Station Type	For OSTN only. Specifies the Station type code as follows: Basic with SP Ratio Cascade Basic without SP Basic with Bias	0

## Notes:

- 1) A line for each DEVICE block should be declared in the CSV file otherwise they will automatically be created given the first field in the Block Name. For example Block name of CIU4.Unit1.Block1 will automatically create a DEVICE block with the name of CIU4 at port COM1: if a line for that device block was not already declared within the file.
- 2) If the DEVICE block name is missing within the block name field #1 then a DEVICE block will be automatically created with the default name of "Device1". Thus you can simply enter a name for each block and they will be added under the "Device1" device. You can then change the name of the device block in the configurator. For example, if PID1024 is specified for field 1 then the driver will place it under the "Device1" DEVICE block.
- 3) Groups should not be included within the CSV file. For example, if there are 200 AIL blocks that need to be imported into the configuration and you wanted to group them you may put them in a single group called AREA1, for example. Block Names such as "CIU4.AREA1.AIL1" to "CIU4.AREA1.AIL200" would be found in the CSV file. However you do not need to add a line for the AREA1 group itself such as "CIU4.AREA1". The group will be automatically created.
- 4) If the logic state descriptors associated with digital blocks are not defined they will be defaulted to values of "Off" and "On".

Example1: File <OPCUnit1.csv>:

Importing the following CSV file will cause the configurator to create a default device block called "Device1". All the the blocks defined within this file will then be placed under that DEVICE block. Note fields 5 to12 are omitted.

```
1BLKULD,DIL,1.1.4.6003,BLOCK UNIT LOAD DEC,Reset,Set,1000
1BLCKTURBD,DIL,1.1.4.6004,BLOCK TURBINE DEC,Stopped,Running
AFDRSPD,STN,1.1.4.6009,A FEEDER SPEED STATION
1BIDINDEV,DIL,1.1.4.6029,B ID FAN INLT DMPR DEV,Off,On
CFDRSPD,STN,1.1.4.6061,C FEEDER SPEED STATION
1CFDRBCKE,RCM,1.1.4.6068,C FEEDER RUNBACK,BAD,GOOD,Stopped,Run
UNIT_GROSS_MW,AIL,1.1.4.663,UNIT GROSS MW
UNIT_GROSS_MVAR,AIL,1.1.4.664,UNIT GROSS MVAR
GEN_VOLTS,AIL,1.1.4.665,GENERATOR VOLTS,0,0.5
```

Example2: File <OPCUnit2.csv>:

Importing the following CSV file will cause the OPC90 Server configurator to create a DEVICE block called CIU4 configured for port COM2:. All subsequent lines in this file will declare blocks to be added to that device.

```
CIU,DEVICE,COM1:,Test
INICI03,100,0,1000,4,10,1000,0,0,0,0,0,0,0,0,0,1,0,10,4,0,0,1,1000,7,0,1,0,3660,0,0,0,
1,0,0,100,4
CIU4.AIL1,AIL,1.1.5.31,,0,0,0,0,0,0,0,0,0,500,1.0
CIU4.AOL1,AOL,40,Test AOL Block,10,13.4,2,-1000,2000,1.5,900,10
CIU4.ODD1,ODD,41,Test ODD Block,60,0,1
CIU4.BLK1,BLK,1.1.3.86,,0,0,0,0,0,0,0,0,0
CIU4.DAANG1,DAANG,1.1.3.48,,0,0,0,0,0,0,0,0,0
CIU4.DIL1,DIL,1.1.3.155,,0,0,0,0,0,0,0,0,0,500
CIU4.MODSTAT1,MODSTAT,1.1.5,,0,0,0,0,0,0,0,0,0
CIU4.POLL1,POLL,1.2.3.4,Poll Block #1,2134
CIU4.SPEC2,SPEC,1.1.3.86,Spec Block #2
CIU4.STN1,STN,1.1.3.86,Station #
```

## 10 Setting up OPC Client Access of OPC90

The RoviSys website ([www.rovisys.com](http://www.rovisys.com)) contains additional documents that provide specific instructions on how to setup OPC90 to work with a variety of commercially available OPC clients. If your OPC client is not listed, please consider writing a short "Using OPC90 With ..." document and submit to [spafford@rovisys.com](mailto:spafford@rovisys.com).

Windows DCOM must be properly configured to allow OPC clients running on the local and / or remote PCs to gain access of OPC90. The Windows program called DCOMCNFG.EXE is used to configure these DCOM settings. This program can be started using the OPC90 Utilities menu item.

Note that depending on the operating environment (Windows Workstation or Server, local user account or domain user account), the exact DCOM settings may vary. The following example sets up the OPC90 settings to "default" which means use those setup for "My Computer". It also assigns the most open settings for "My Computer". If this example does not work for your system or tighter security is required, try different combinations until one is found that works correctly and satisfies the security requirements. For example instead of setting OPC90 to use the "My Computer" default settings, assign it custom settings with a smaller set of authorized users. Running OPC90 as a service will sometimes require additional adjustments depending on the operating system in use, its workgroup / domain residency and whether or not the service is running as a system service or preferably (and required when using the OPC90 "Database Shadowing feature") in a privileged user account. Note that when running as a service, make sure it is stopped and started after any changes to the DCOM settings are made.

The local OPC90 PC and remote PCs that will be running OPC client software must both be setup. The settings presented in this section provide the most open remote OPC client access to OPC90 Server. Once remote access has been setup and validated, the settings can be adjusted for any site specific security needs.

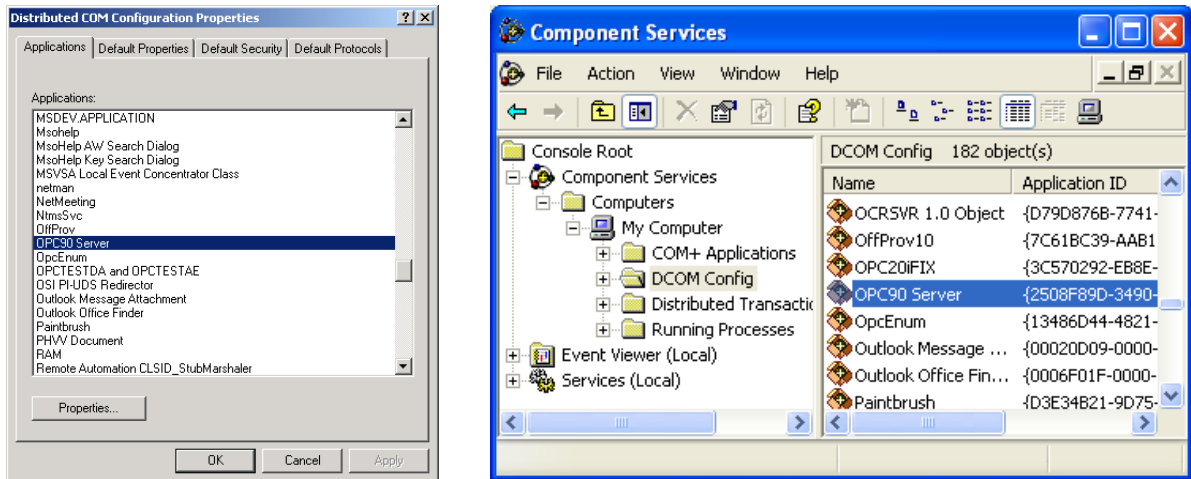
### 10.1 Configuring DCOM On The Local OPC90 PC

The first step is to select or setup a user account that at a minimum is a member of the "Power Users" group but preferably the "Administrator" group. This account will be used by DCOM to run OPC90 as a service or when an access request is received from a remote OPC client. If such an account does not exist, create one called "OPC90", assign it to the "Administrator" group and assigned a valid Windows password to the account such as "opc90server".

When both local and remote PCs have been setup to exist within a domain, both should use the same domain user account to run OPC90 and the clients that will attach to it. If the PCs are setup in a workgroup, the user account and password selected to run OPC90 on the local PC must be duplicated on the remote PC and the OPC clients on the remote run under that same account.

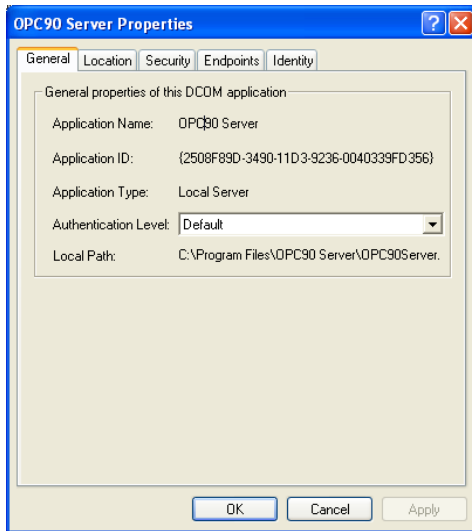
If Windows XP / 2003 Server is being utilized with Service Pack 2 or later and the Windows firewall is enabled, OPC90 and the OPC clients must be added to the list of firewall exceptions for each PC running OPC90 or the OPC client. Also a port named DCOM with port number 135 associated with TCP must be added to the local and remote PCs. This is setup by running ControlPanel | Windows Firewall. For additional information consult the document entitled “OPC With DCOM With XP SP2.PDF”. This document has been installed in the C:\Program Files\OPC90 Server directory.

To run DCOMCNFG select “Start | Run” and type in DCOMCNFG (or start it from the OPC90 Utilities menu selection). Based on the Windows operating system being used, one of the two following program windows will be displayed.

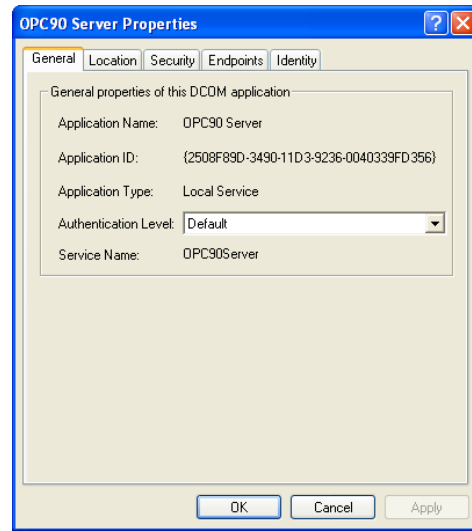


Select “OPC90 Server” which can be found under the Application tab list box or under “Components | Computers | My Computer | DCOM Config”. Click on the properties button (or right click on “OPC90 Server” and select properties) and one of the two following dialogs will appear.

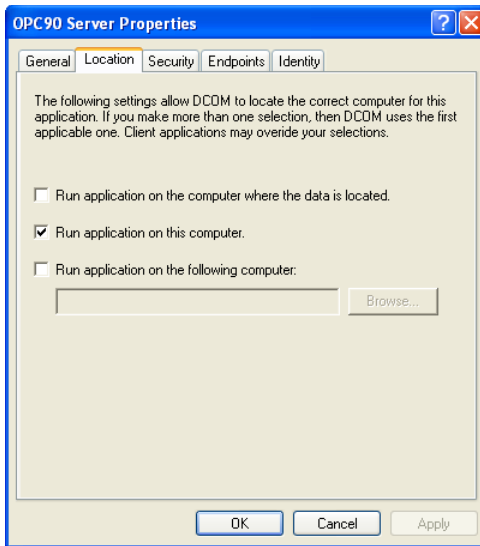
OPC90 not as a Service



OPC90 as a Service

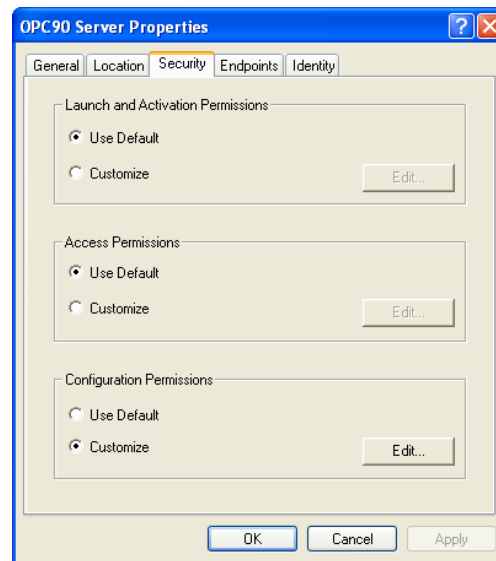


Under the General tab set the Authentication Level to (Default). Click on the location tab and the following dialog will appear:



Enable the “Run application on this computer” checkbox.

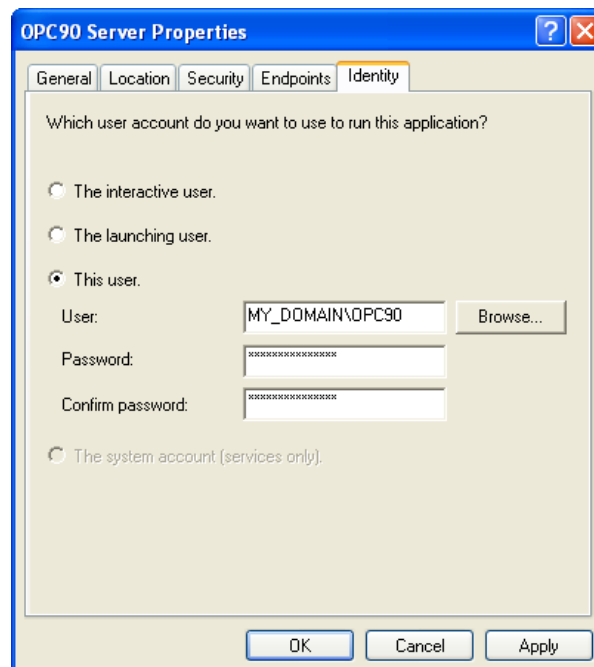
Click on the Security tab and the following dialog will appear:



Make the following selections:

- Use default access permissions,
- Use default launch permissions,
- Use customized configuration permissions

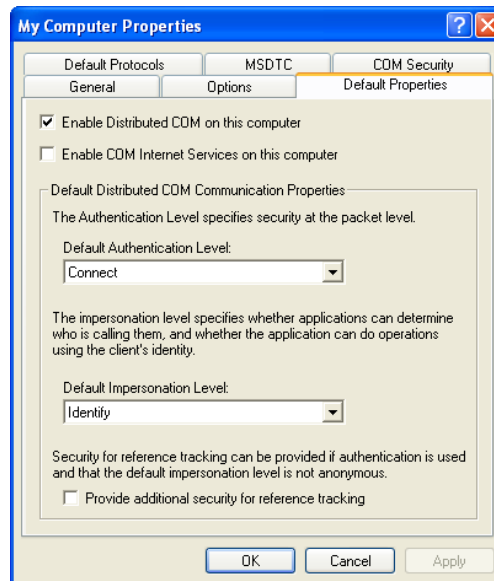
Click on the Identity tab and the following dialog will appear:



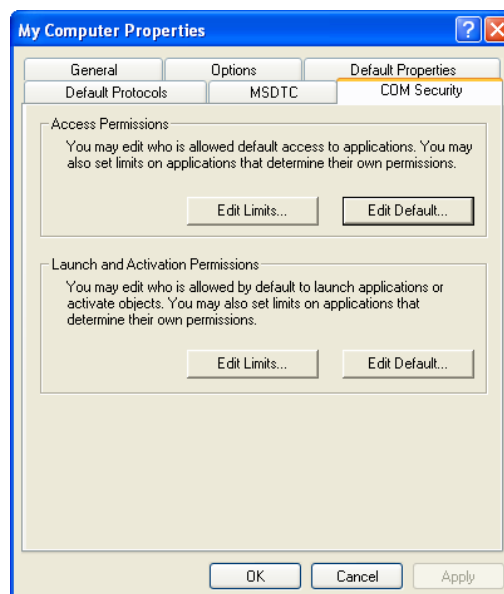
Select *this user* and type in the account information that will be used to run the OPC90 application. The account information must include the domain controller computer name (or local PC name for workgroup based systems), backslash character and user account

that will be used to run OPC90. This example shows “MY\_DOMAIN\OPC90”. Use the actual domain or workgroup computer name and user account name for the specific system OPC90 has been installed.

Next select the DCOM properties of “My Computer” and set the following default properties:



Select the COM Security tab shown as follows:



Click on “Edit Default” for both the Access Permissions and Launch and Activation Permissions and **add the following list of users**. Make sure “allow access” for both local and remote is enabled for all of these users. For Windows XP SP2 and Windows



2003, also check Edit Limits options for both the Access Permissions and Launch and Activation Permissions and make sure that these accounts are also added.

## ANONYMOUS LOGON

Everyone

INTERACTIVE

NETWORK

NETWORK SERVICES \*

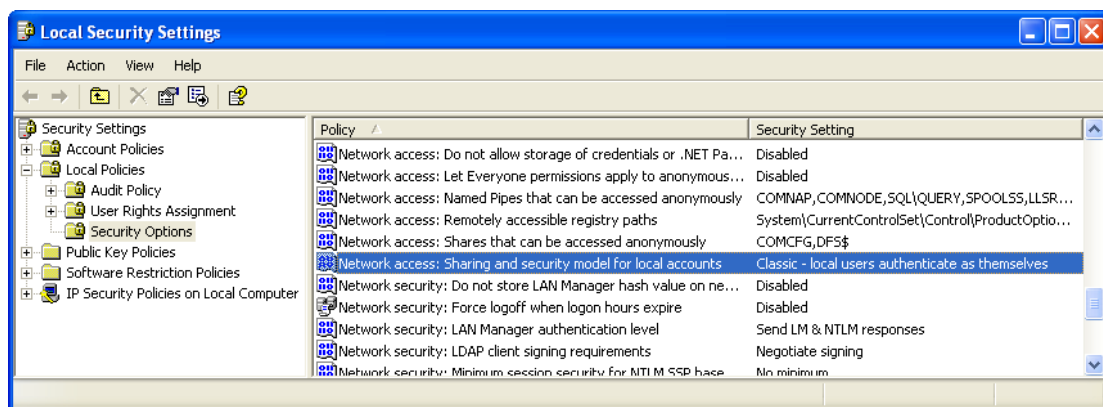
Self \*

SYSTEM

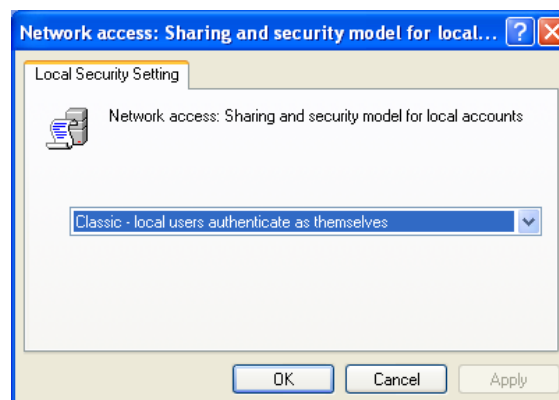
Distributed com users\*

\* These users are not supported on earlier Windows operating systems.

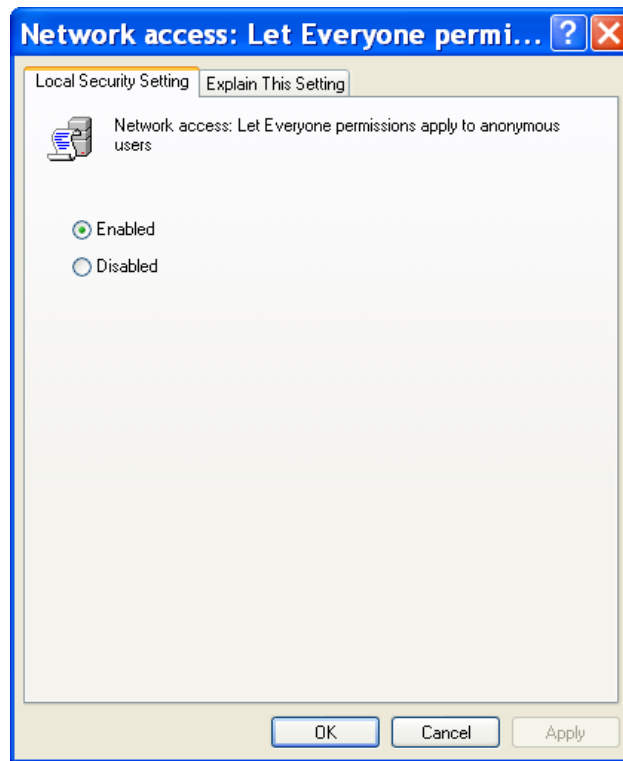
There are two local security options policies that need to be setup. Run Control Panel | Administrative Tools | Local Security Policy and select “Security Options” as displayed by the following.



Double click on “Network access: Sharing and security model for local accounts” and set it to “Classic” as shown by the following dialog.



Double click on “Network access: Let Everyone permissions apply to anonymous users” and set it to “Enabled” as shown by following dialog.

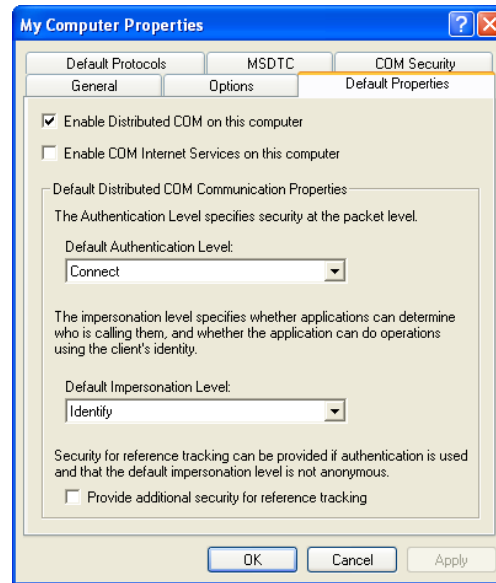


Setup of DCOM on the local OPC90 PC is complete. If the OPC client cannot access OPC90 after completing this setup, try rebooting the PC. Depending on the types of changes made to "My Computer" DCOM settings a reboot of the PC may be required.

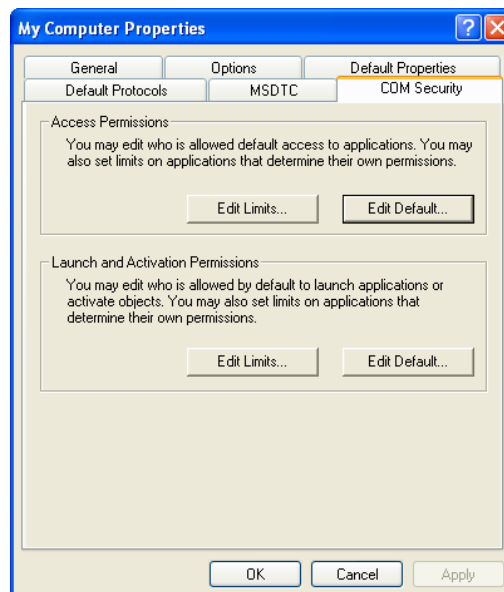
## 10.2 Configuring DCOM on the Remote OPC Client PC

Some OPC clients utilize a mechanism called “Data Callbacks” to receive data values from an OPC server. For these types of clients, the remote PC must have its access permissions enabled to receive the callbacks. If an OPC client can attach to the server but not receive data values, this is a sign it relies on callbacks and therefore the PC access permissions must be adjusted. Do the following procedure to make these adjustments.

Run DCOMCNFG and select the DCOM properties of “My Computer” and set the following default properties:



Select the COM Security tab shown as follows:



Click on “Edit Default” for the Access Permissions and add the following list of users. Make sure “allow access” for both local and remote is enabled for all of these users.

## ANONYMOUS LOGON

**Everyone**

**INTERACTIVE**

**NETWORK**

**NETWORK SERVICES \***

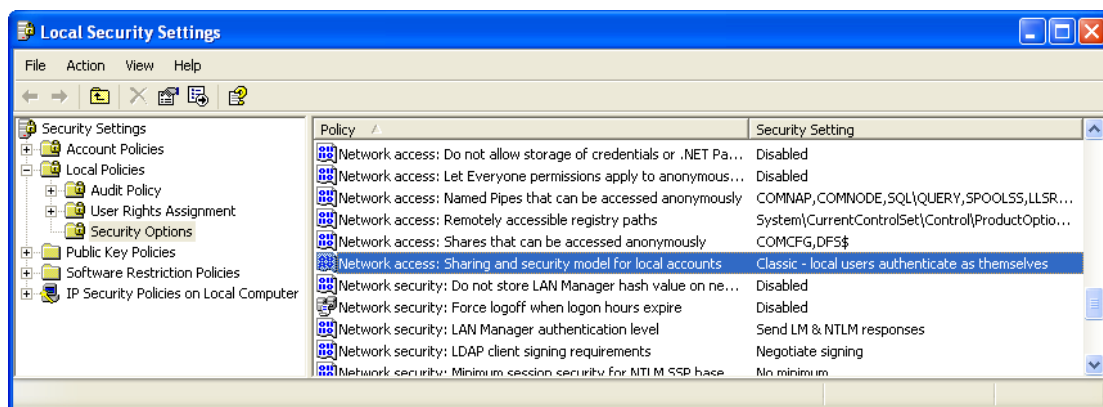
**Self \***

**SYSTEM**

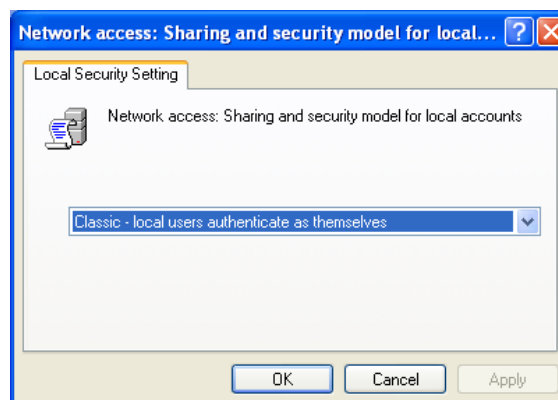
**Distributed com users\***

\* These users are not supported on earlier Windows operating systems.

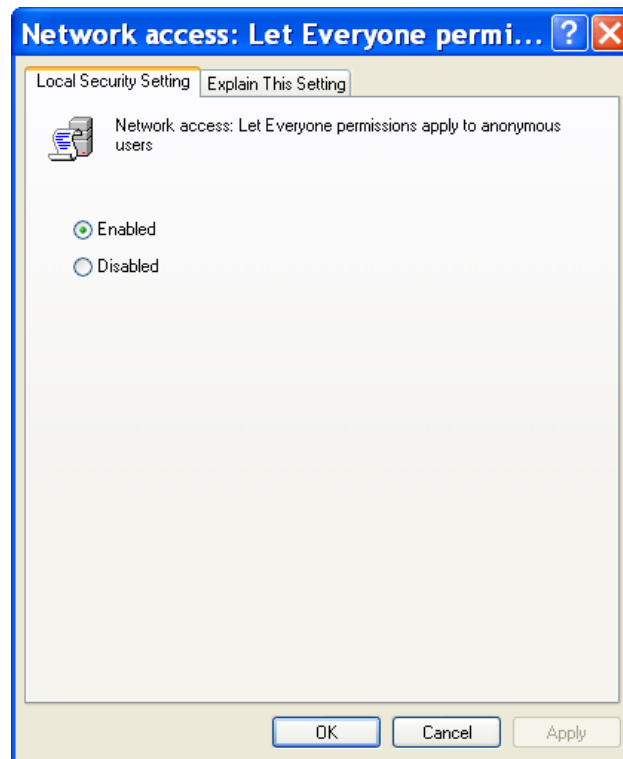
There are two local security options policies that need to be setup. Run Control Panel | Administrative Tools | Local Security Policy and select “Security Options” as displayed by the following.



Double click on “Network access: Sharing and security model for local accounts” and set it to “Classic” as shown by the following dialog.



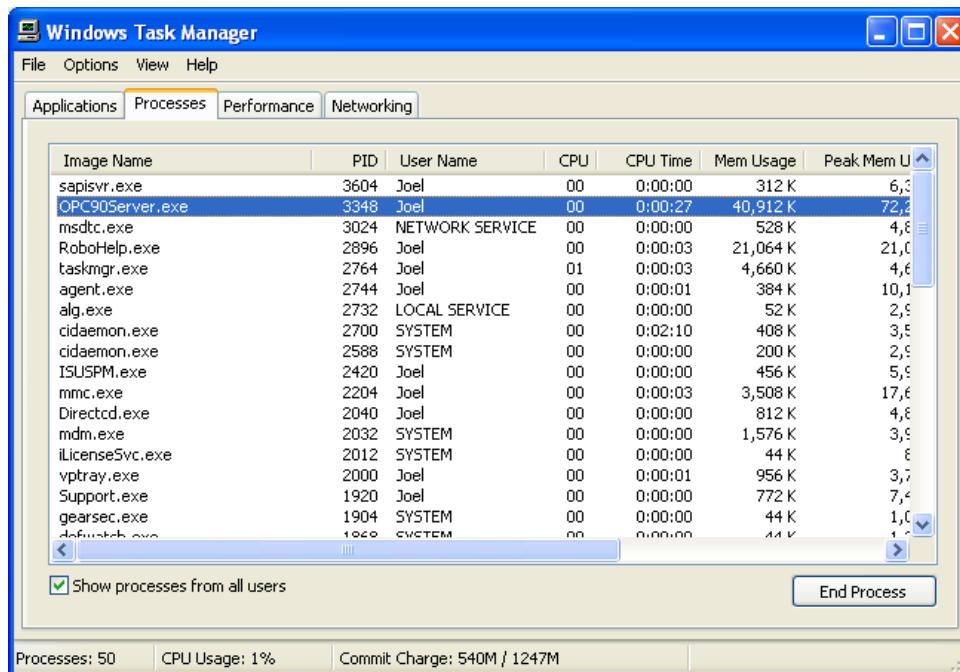
Double click on “Network access: Let Everyone permissions apply to anonymous users” and set it to “Enabled” as shown by following dialog.



Setup of DCOM on the remote PC is complete. If the OPC client cannot access OPC90 after completing this setup, try rebooting the OPC client PC. Depending on the types of changes made to "My Computer" DCOM settings a reboot of the PC may be required.

### 10.3 Validating OPC Client Access of OPC90

When one or more OPC clients are started and they have requested a connection with OPC90, a single instance of the program can be observed using the Processes tab of Windows Task Manager shown by the following dialog.



If OPC90 has been configured to run as a service, a single instance of the program should be observed prior to running any OPC client.

If multiple instances of OPC90 are found running after two or more OPC clients have been started (and OPC90 has not been manually run), this is an indication of a DCOM setup error. This condition must be corrected before OPC90 can be properly utilized to serve its data to more than one OPC client. Failure to do so will result in only the first OPC client to run receiving data from OPC90.

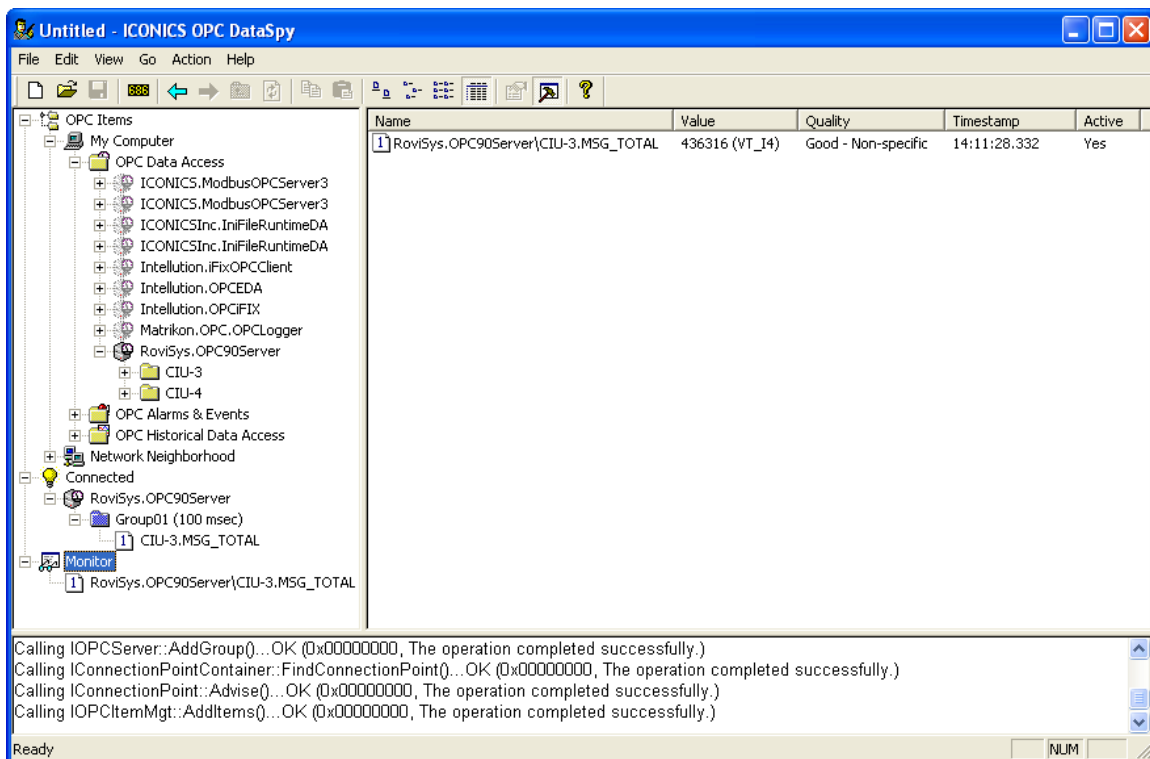
Manually running OPC90 after it has been started as a service or by an OPC client connection will result in a second instance of the program to be observed in Windows Task Manager. This is by design and not an indication of an error condition. Under this condition a progress dialog will be briefly displayed when it is first started, reporting that it is attaching itself to the first instance. Afterwards the second instance becomes a user interface to the first instance.

**Warning**, manually running OPC90 should not be attempted until after all the OPC clients have been started. Failure to heed this warning could result in an OPC client connecting with the second instance of OPC90 that is only serving as a user interface to the first instance. Under this situation, that OPC client will not receive the data being collected by the first instance.

Installation of OPC90 includes two different OPC test clients. They are useful for validating local and remote PC OPC client connection with OPC90. The first test client is called OPC Expert. It requires Microsoft .NET 3.5 framework to be installed on the PC. Double click on "C:\Program Files\OPC90 Server\OPC Expert.exe"

(or “C:\Program Files (x86)\OPC90 Server\OPC Expert.exe”) to run it. The second OPC test client is called DataSpy. It must first be installed before it can be used. Double click on “C:\Program Files\OPC90 Server\opc\_datasp901.exe” (or “C:\Program Files (x86)\OPC90 Server\opc\_datasp901.exe”) to install it. After installation, run it from Start | Iconics | OPC Data Spy | OPC DataSpy.

If OPC client connection problems occur, try establishing an OPC connection using the DataSpy OPC client shown by the following dialog. Select “View | Output” so that it will post OPC messages to a list view box at the bottom of its program window. These messages give sufficient details to determine successful / problem operation.



For example if an error is posted for the IConnectionPoint::Advise() call, that would be an indication that the OPC90 PC does not have sufficient access privileges to setup call back functions within the OPC client PCs address space. This would indicate a DCOM or user account setup error. Review the DCOM settings. Also remember that if the local and remote PCs are members of a domain, make sure the user account that is running OPC90 and the OPC clients have sufficient privileges (Power Users minimum or Administrator). If the PCs are members of a workgroup, they both must run OPC90 and the OPC clients from user accounts that are the same on both PCs.

## 11 Using OPC90 with IET800

Several years ago ABB released an Ethernet based CIU. For Symphony Plus type systems it is generically reference as SPIET800 and INFI 90 type systems, ICI800. Regardless of system, the Ethernet CIU is comprised of two module types, which are IET800 and NIS21. This document references the device as IET800.

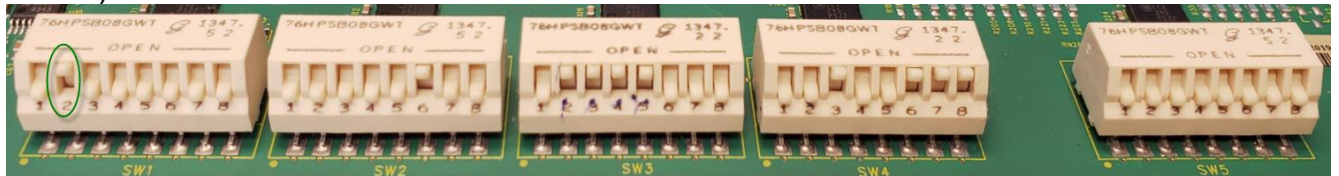
The IET800 capabilities matched its SCSI predecessor CIU type. The differences being replacement of the SCSI port with two Ethernet ports (only one is functional). The original IET800 design included two RS232 serial ports that can be set for a maximum rate of 115,200 baud. A switch setting determines if IET800 operates in Ethernet or serial mode. Simultaneous operation of both not supported.

After 38 years of providing multiple generations of CIU devices that are serial communication capable, ABB has decided serial ports are no longer secure. ABB has stated IET800 boards manufactured after August of 2019 will have the serial ports disabled. Pretty bizarre ABB decision, considering a serial port is much more secure then the Ethernet port, reachable by any device on the network.

OPC90 supports communication with the IET800 Ethernet port as well as high speed serial ports for boards that still have them.

### 11.1 IET800 Switch Setup

Setup the IET800 switch settings as follows (see ABB IET800 instruction manual for details).



SW1 All Poles = 0 (closed or on) except Pole #2.

SW1 Pole #2 enables **Ethernet** when 1, **serial** when 0

SW2 Pole #6 = 1 (open or off), enable Cnet/INFI-NET diagnostics.

SW2 Pole #1, 2, 3, 4, 5, 7, 8 = 0 (closed or on).

SW3 Pole #1 = 0 (closed or on).

SW3 Pole #2, 3, 4, 5 = 1 (open or off), sets both serial ports to 115,200 baud.

SW3 Pole #6, 7 = 0 (closed or on), sets serial ports for 8 data bits, 1 stop bit, no parity.

SW3 Pole #8 = 0 (closed or on), enables 2<sup>nd</sup> serial port for computer communication.

SW4 Pole #3 = 1 (open or off), enables port checksum option (required).

SW4 Pole #1, 2, 4, 5 = 0 (closed or on), ABB required for normal operation.

SW4 Pole #6, 7, 8 = 1 (open or off), ABB required for normal operation.

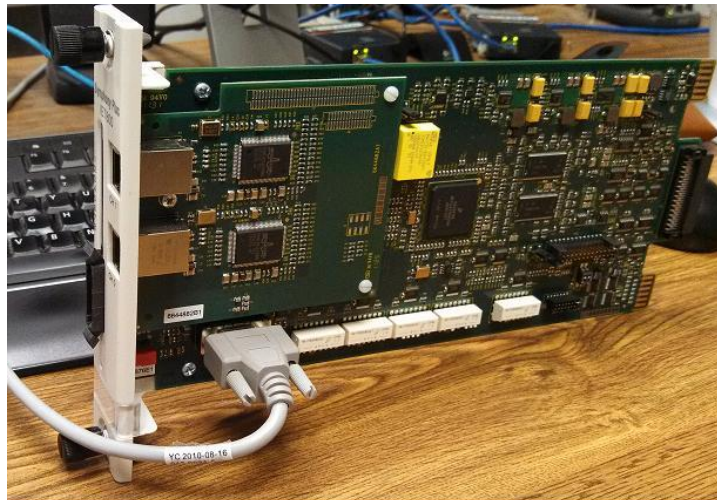
SW5 All Poles = 0 (closed or on), ABB required for normal operation.



## 11.2 IET800 Parameter Setup

OPC90 requires IET800 firmware revision A7 or later for Ethernet communication. Earlier version IET800 firmware is OK for serial communication. RoviSys has tested serial communication with IET800 firmware A6 and A7. It may work with some earlier version firmware but if it indicates a security issue, IET800 firmware A7 needs to be loaded. Customers who have purchased ABB support can log into the ABB website at [www.abb.com/User/Login.aspx](http://www.abb.com/User/Login.aspx) and download the firmware. The IET800 user manual includes a section explaining how to load the firmware, easily accomplished using Microsoft Windows Hyper-terminal program or free Tera Term program.

Installation of the IET800 requires a Basic Security License key purchased from ABB. The ABB part number for the license is SSBIET80000000. The software license key can be loaded manually by attaching a terminal (such as Microsoft Windows Hyper Terminal or Tera Term) to the 9 pin serial port directly on the IET800 circuit card. A straight through serial cable is required for this connection. The cable needs a 9 pin male connector on one end and 9 pin female connector on the other end. It is important to note, the IET800 on board serial port **isn't** intended for communication with OPC90. This port is a utility port used to load and check the software license and its security status.



This image shows a RS232 straight through cable attached to the IET800 on-board serial port. The other end of the cable plugs into a PC COM port and Windows Hyper-terminal or Tera Term attached to that port (**@ 9600 baud**). When attached correctly the following menu is displayed.

```

ABB Inc. - IET800 Firmware Revision A_6.005
Copyright - 1998-2015 by ABB Inc. All Rights Reserved
1-->TALK90
2-->Configure/View callup password
3-->Computer command/reply sequences (OFF)
4-->Set local time/date
5-->Monitor mode (Disabled)
6-->Real Value Conversion
7-->Port address (-1)
9-->Default Time Sync accuracy (3)

```

Select option: \_

The diagnostic sub-menus are reached from this menu by pressing the “0” key. The following line is displayed.

IET800 Diagnostic Utilities Revision A\_6.05

Press the “P” key displays the following.

```

IET800 Diagnostic Utilities Revision A_6.05
NOTE: set DIAGNOSTIC LEVEL > 0 (option 'L') to get messages
Current PRINT FLAG = 0
error messages = 1
local messages = 2
menus          = 4
remote messages = 8
Enter new PRINT FLAG (HEX): _

```

Press the “F” key to select all of these messages and menus, which will display the following sub-menu.

```

IET800 Diagnostic Utilities Revision A_6.05
1--> LOCAL NIS/LIS REQUEST          2--> REMOTE NIS/LIS REQUEST
3--> LOCAL MODULE UTILITIES        4--> REMOTE MODULE UTILITIES
5--> ERROR MESSAGE MONITOR          7--> READ LOOP TOPOLOGY
B--> BUILD A MESSAGE                T--> SET TERMINAL TYPE
C--> ICI/CIU UTILITIES              P--> PRINT FLAG (=f)
U--> CPU UTILIZATION                Y--> SET YEAR-DATE-TIME
SELECT OPTION: _

```

Select “3” (Local Module Utilities) to display the following sub-menu.

#### Local Module Utilities

```

1--> READ/WRITE MEMORY
2--> EXAMINE BLOCK RECORDS
3--> EXAMINE NODE RECORDS
4--> DIAGNOSTIC COUNTERS
5--> MEMORY UTILIZATION
6--> REAL VALUE CONVERSION
7--> EXCEPTION STATISTICS INITIALIZE
8--> EXCEPTION STATISTICS READ
9--> RED LIGHT HALT ANALYSIS
E--> ELECTRONIC IDS
F--> FAILOVER STATISTICS READ
M--> READ MODULE DETAILS
N--> READ NODE PERFORMANCE STATISTICS
P--> READ REDUNDANCY LINK STATISTICS
R--> ROS TASK STACK USE
S--> TCP/IP UTILITIES MENU
T--> READ TIME SYNC MASTER ID

```

SELECT OPTION or press ESCAPE to exit: \_

Select “E” (ELECTRONIC IDS) displays the following sub-menu.

```

ELECTRONIC IDS
0--> Read all MicroLAN Electronic IDs
1--> Read License
2--> Enter License
3--> Reset License
4--> Read MAC Address
5--> Read/Enter IP Address/Mask/Gateway
6--> Read License/Security Status
7--> Test License

```

Enter option:

The ELECTRONIC IDS sub-menu is used to finish setup of the IET800.

Selection 2 allows entry of the Basic Security license purchased from ABB. The license **must** be enabled on the IET800 before using it to access the system. It is bound to the MAC0 address of the IET800. **Note** the calculation of the MAC address was changed with the introduction of IET800 firmware revision A5 and later, so it may not match the hardware ID sticker on the back of the module. Use selection 4 to read the MAC0 address should ABB request it for generation of the software key needed to enable the Basic Security license. The software key document received from ABB looks as follows.

Software Key					
Last Update: 2016-Jan-14			License Id: SL644222721245164		
			License Type:		
			Version: 1		
ABB Contact			End User		
ABB Inc.			Customer Name		
29801 Euclid Avenue			1234 Anywhere Street		
N/A					
OH			Honolulu, Hawaii 96818		
Wickliffe, N/A 44092-1832			United States		
United States					
Attn.: VirtualAssistant.usiny@us.abb.com			Attn: George Washington		
Phone: 440-585-8820			Phone: 440-585-8500		
Fax: 440-585-5087			Fax:		
Product name: IET800, vA.0					
Product version: A.0					
Hardware Id: C43CD312E1EE					
Hardware Id Type: Ethernet					
Feature	Ver	Users	Software Key	User String	Expire Date
IET800	A.0	0	3E5C71B63D6B78217424	""	01-jan-00

The “Software Key” provided in this document (3E5C71B63D6B78217424 for this example) is the 20 digit value entered using selection 2. Correctly enter software key to enable the ABB Basic Security license. The IET800 user manual indicates failure to enable the license will result in the IET800 randomly resetting itself. This would interrupt flow of process data, obviously a potentially dangerous situation to avoid. Selection 6 provides information regarding the potential for this occurring.

Selection 5 allows display and entry of the IET800 IP address information needed to communicate using Ethernet. Important, the IP address **must** be setup even if using serial communication with OPC90. Setup an IP address that is consistent with the IP addresses utilized on the network the OPC90 PC and IET800 are connected. The IP mask (typically 255.255.255.0) and correct network Gateway IP address **must** be defined. Contact the company IT group if these values are unknown.

Selection 6 displays the IET800 license and security status. If it reports the IET800 license isn’t valid, it **must** be entered (see selection 2). If it reports the security reset grace period is active or the security status is failed (*red box added for emphasis*), OPC90 may not be able to establish communication with the IET800. Make sure the

IET800 IP addressing information has been setup before attempting to make the reset grace period inactive. If the security reset grace period is inactive and the security status is not failed, no further steps are required and the IET800 is ready for communication with OPC90. If not, read on.

```
License and Security Status for IET800 module:
This module has a valid IET800 product license
The PSI indicator says this module is primary
The security reset grace period is active
The SFF indicator says security status is failed
The LFF indicator says the license is not failed
The BMR indicator says the backup module is missing or stopped
The BTR indicator says the backup module is not takeover ready
```

New IET800's received from the factory are typically in this state. Note that using the ABB Harmony API Configuration program will leave the IET800 in the failed state. This includes both Ethernet and serial communication. **Note** also when using ABB Harmony API Configuration program to communicate serially, it will launch a process called DEVICE.exe that handles serial communication with the IET800. After exiting Harmony API Configuration program, run task manager and end the DEVICE.exe process before doing the following activity.

From the Electronic IDS menu, select **3** to reset the license.

```
ELECTRONIC IDS
0--> Read all MicroLAN Electronic IDs
1--> Read License
2--> Enter License
3--> Reset License
4--> Read MAC Address
5--> Read/Enter IP Address/Mask/Gateway
6--> Read License/Security Status
7--> Test License
Enter option: 5
```

The following scary warning is displayed. It is safe to proceed by pushing the E key.

```
WARNING ***** Reset of License may make IET800 Inoperable:
Hit 'E' to reset license, ESC to exit:
```

After it indicates the license reset is complete, wait at least 30 seconds watching the IET800 8 LEDs. They will all simultaneously blink on and off 3 times (assumes IET800 firmware revision A6 and A7). After this occurs, push the IET800 reset button to stop it and then push it again to start it back up. Watch the 8 LEDs and you will see all of them blink simultaneously 6 times. This is the indication the license is ready for use by OPC90. It can be verified by returning to the Electronic IDS sub-menu and selecting **6** to read the license and security status. It will display the following (*green box added for emphasis*).

```

License and Security Status for IET800 module:
This module has a valid IET800 product license
The PSI indicator says this module is primary
The security reset grace period is inactive
The SFF indicator says security status is not failed
The LFF indicator says the license is not failed
The BMR indicator says the backup module is missing or stopped
The BTR indicator says the backup module is not takeover ready
    
```

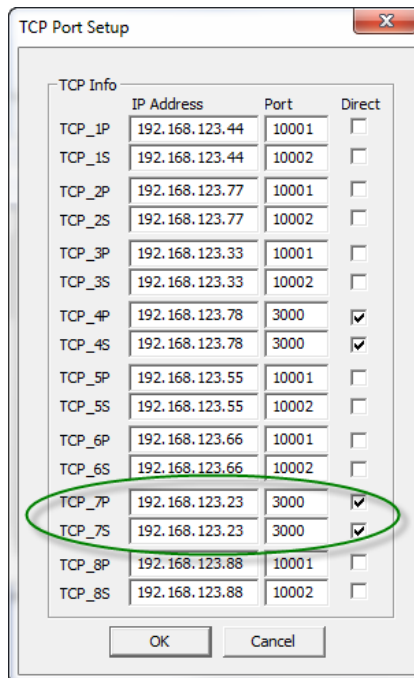
Notice the security reset grace period is inactive and the security status is not failed. This means the IET800 won't do the random reset, mentioned in its user manual. **Important**, at this point the ABB Harmony API Configuration program shouldn't be used to communicate with this IET800. Otherwise, it will reactivate the security reset grace period and set the security status to failed requiring repeat of this procedure.

### 11.3 IET800 Ethernet Port Setup

When communicating Ethernet, attached the IET800 channel 0, RJ45 jack to the network. Channel 1 is not supported. From OPC90 PC, run a command prompt and ping the IET800 IP address. This verifies IET800 network attachment and addressed correctly.

### 11.4 OPC90 Ethernet Communication Setup

Select OPC90 Edit | TCP Ports menu item. Enter the IET800 IP address in a pair of TCP ports (TCP\_7P and TCP\_7S in this example). Set the port number for both of these IP addresses to 3000. Click the Direct check box next to the two ports. This specifies OPC90 will directly communicate with the IET800 Ethernet port.



Right click on OPC90 DEVICE block and select properties. Set "Redundant Channel" communication scheme. Set primary port to TCP\_7P (per this example) and secondary port to TCP\_7S. OPC90 sets up two Ethernet socket connections with the IET800. First

socket used for exchanging exception report data. The other socket used for exchanging polled data. Save configuration and OPC90 / IET800 should be ready for Ethernet communication.

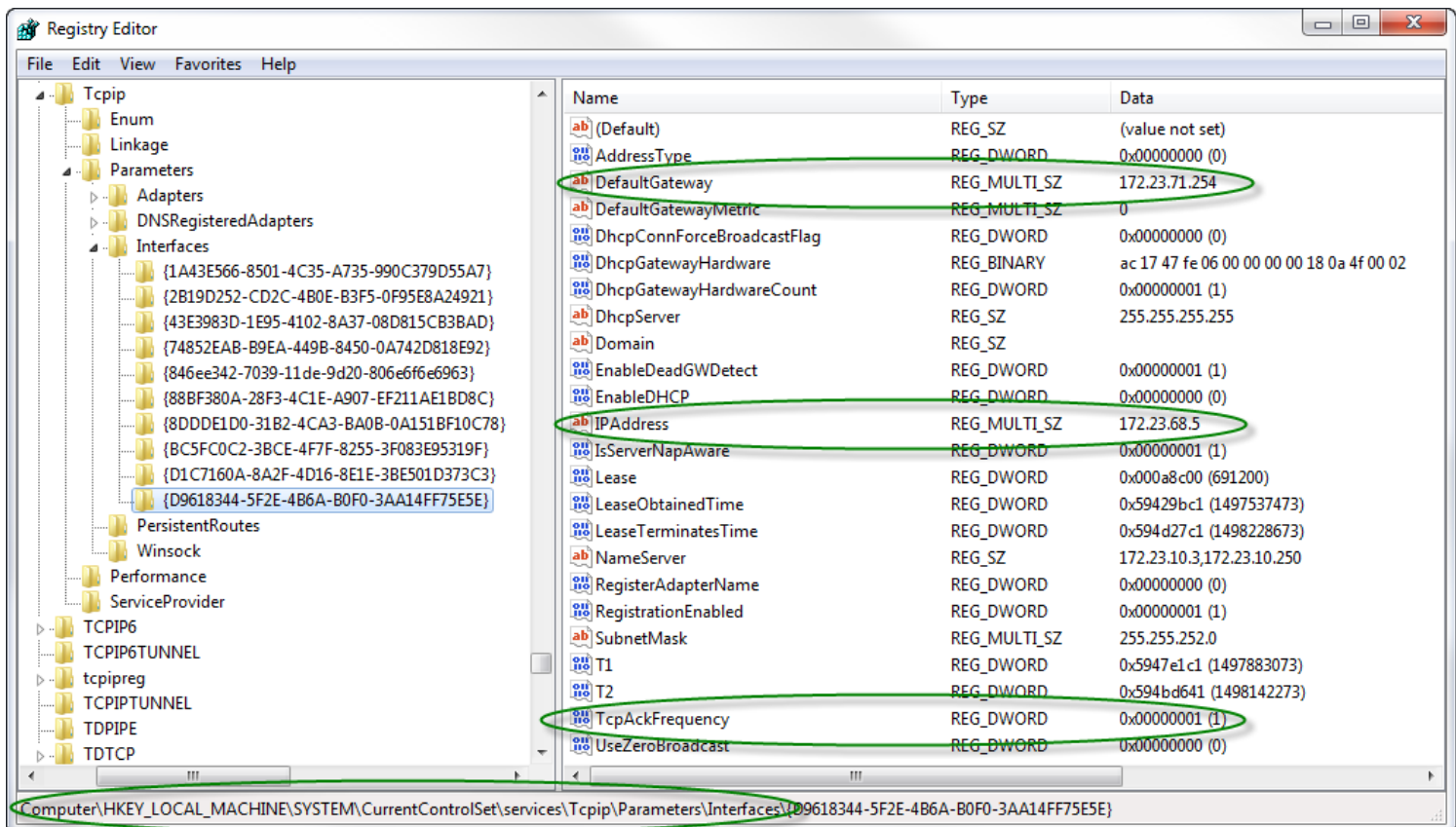
To complete setup, one registry entry must be added to OPC90 PC to maximize communication throughput. OPC90 automatically makes the registry entry and request user to reboot PC to make it take effect. **Failure to add this registry setting results in very slow communication performance.** If this is observed, follow this manual procedure to verify OPC90 has successfully added the key. If not, it needs to be manually added.

From a command prompt (run cmd) enter IPCONFIG. It will display a list of LAN adapters along with the Ethernet adapter for the Local Area Connection. It looks like this.

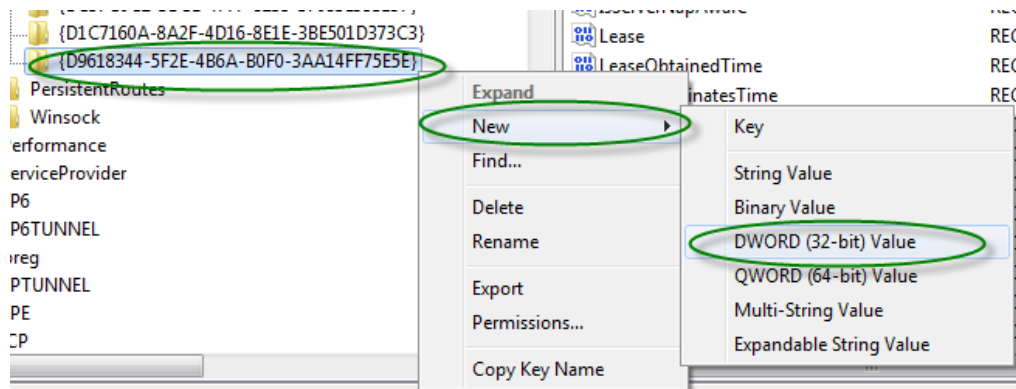
```
Ethernet adapter Local Area Connection:
Connection-specific DNS Suffix . :
IPv4 Address. . . . . : 172.23.68.5
Subnet Mask . . . . . : 255.255.252.0
Default Gateway . . . . . : 172.23.71.254
```

For this example, the OPC90 PC IP address is 172.23.68.5 and Default Gateway is 172.23.71.254. Use this information to find the correct location to add the needed registry key. Run REGEDIT and drill down to “Computer\HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Interfaces” registry area.





Click on each of the Interfaces until you find the one that contains the Default Gateway IP address listed by the IPConfig command (in this example 172.23.71.254) and PC IP address (172.23.68.5). Right click on that interface key and select New | DWORD (32-bit) Value.



The name of this new value must be "TcpAckFrequency" and it **must** be set to 1. Reboot the PC and OPC90 is now ready for Ethernet communication with IET800.



## 11.5 IET800 Serial Port Setup

Skip the setup presented in this section if using Ethernet communication. The IET800 needs attached to a NTMP01 serial termination unit. Appendix B of its user manual explains that connection. A Process Bus Adapter (ABB part # P-HC-BRC-PBA20000) is attached to the back of the module mounting unit slot where the IET800 is plugged in. The hooded end of the NKTU01 cable plugged into the back of the PBA. The other end is plugged into the NTMP01 P1 (may also be labeled MFP1) connector. The IET800 is now ready for high speed single or dual channel RS232 serial communication.



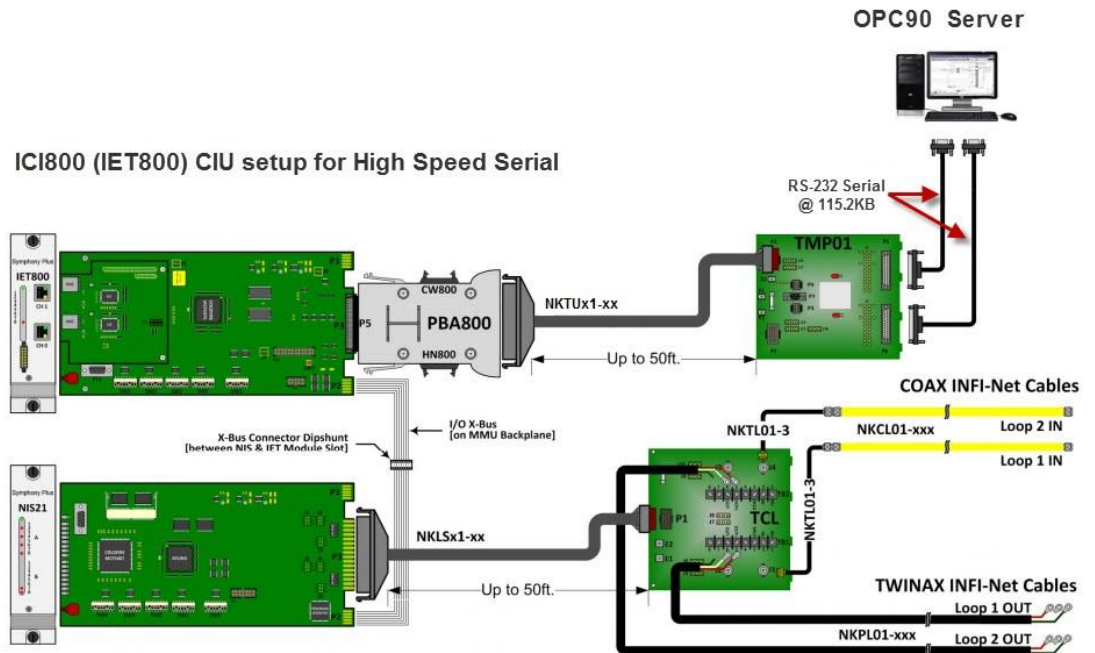
Process Bus Adapter  
(PBA)



NKTU01 into PBA back of  
MMU



NKTU01 – PBA – IET800



Block diagram shows setup for all ABB hardware components.

## 11.6 OPC90 Serial Communication Setup

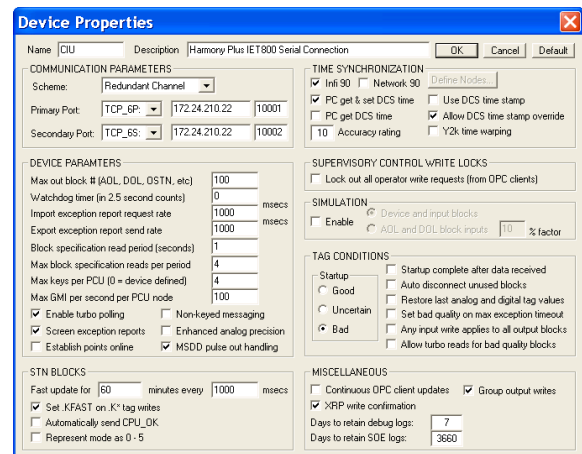
Skip the setup presented in this section if using Ethernet communication. The OPC90 DEVICE block should be set to the “Redundant Channel” communication scheme (not absolutely necessary but will provide the highest possible data update rates). This means it will communicate with the IET800 using two serial ports. The two serial ports can be physical ports on the PC running OPC90 or those associated with an Ethernet to Serial device. If using physical COM ports those ports need to be selected for the OPC90

DEVICE block Primary Port and Secondary Port settings. The speed of those ports must also be set to 115,200 baud using the OPC90 “Edit | Com Ports” menu selection. Cable those PC COM ports to the IET800 NTMP01 24 pin RS232 serial ports typically labeled “Terminal” and “Printer”.

Newer PC hardware typically does not include serial ports. An easy solution for adding serial ports is to utilize an Ethernet to Serial device. The Lantronix UDS 2100 device has been tested for this purpose. It contains two serial ports cabled to the two serial ports on the IET800 NTMP01 termination unit. Another nice appeal for this solution is it allows the PC running OPC90 to be located at a farther distant from the IET800 because the Ethernet network is used for the connection. This allows the Ethernet to Serial device to be located in proximity to the IET800. Note this setup is also the required approach when running OPC90 in a VM environment if direct IET800 Ethernet communication not utilized.

Purchase of Lantronix UDS 2100 includes a software CD (or downloaded from Lantronix website). That CD is used to install the Lantronix CPR Manager and DeviceInstaller software. The DeviceInstaller allows the IP address of the unit to be configured and allows the default baud rate of the two ports to be defined. **The default baud rate must be configured for 115,200 baud.** The CPR Manager allows definition of virtual COM ports. The virtual COM ports appear as real ports to Windows but are actually a mapping of the COM port name to the IP address and port on the UDS 2100 unit. The OPC90 DEVICE block Primary Port and Secondary Port properties can be configured to reference these virtual ports as if they are physical ports on the PC.

OPC90 includes the ability to bypass the Lantronix CPR Manager virtual ports (recommended) and directly reference the UDS 2100 IP address and port within that IP location. This is accomplished by selecting “TCP\_1P” for the DEVICE block Primary Port property and “TCP\_1S” for Secondary Port property. Selection of these ports enables the IP address of the Lantronix UDS 2100 to be entered along with the port number within that IP location. By default, the first UDS 2100 port number is 10001 and the second one is 10002. See the OPC90 help sections entitled “TCP Communication” and “TCP Ports” for more details on this feature.



## 11.7 Parts Summary

- Basic security license ABB part # SSBIET80000000

### Additional Parts Summary for Serial Communication

- Process Bus Adapter ABB part # P-HC-BRC-PBA20000
- PBA to NTMP01 cable ABB part # NKTU01 or NKTU11
- Serial termination unit ABB part # NTMP01
- Two RS232 9 pin female to 25 pin male serial modem cables
- Lantronix UDS 2100 Ethernet to Serial converter (when PC doesn't have 2 serial ports or distance between PC and IET800 exceeds 50 foot)

## 12 Migrating OPC90 to a Windows 64 Bit Environment

OPC90 can run in a Windows 64 bit environment. Follow these steps when migrating from a 32 bit to 64 bit environment.

Before Installation:

- Log into a privileged account.
- Turn off UAC (User Access Control).
  - If you don't know how, do a search from Windows help for UAC and the results will provide a link to do this.
- Run the setup from the OPC90 CD or downloaded installer.

After Installation:

- Using Windows explorer set the "Run as administrator" properties of C:\Program Files (x86)\OPC90 Server\OPC90Server.exe.
  - Right click on this file and select properties. Under the compatibility tab enable "Run as Administrator".
- Copy your current database file from the XP PC to the new 64 bit PC.
  - C:\Program Files\OPC90 Server\CFG to C:\Program Files (x86)\OPC90 Server\CFG.
- Run OPC90 and open the database file (File | Open).
- Set OPC90 to "Startup in Runtime" (View | Startup In Runtime).
- Set OPC90 to run as a service (View | Run as Service).
- Use the Windows service manager (OPC90 automatically will start this up for you) and set the Log On tab of the OPC90 Server services to run from a privileged account
  - This step may not be required if the OPC client and OPC90 are all running on this same PC.
  - Find OPC90 Server service in Windows service manager window, right click on it and select Log On tab.
- Using Windows Explorer find and double click on C:\Program Files (x86)\OPC90 Server\OPC90\_INI\_ADD\_MAPPING.REG and answer yes to allow registry update.
- Run Control Panel | Administrative Tools | Local Security Policy | Security Settings | Local Policies | Security Options.
  - "Network access: Let Everyone permissions apply to anonymous user" must be **Enabled**

- “Network access: Sharing and security model for local accounts” must be set to **Classic – local users authenticate as themselves**
- Run Control Panel | Administrative Tools | Component Services | Computers | My Computer (right click on My Computer and select properties)
  - Select COM Security Tab. Under Access Permissions click on “Edit Default” button.
  - Add these users: “Everyone”, “ANONYMOUS LOGON”, “INTERACTIVE”.
  - Select COM Security Tab. Under Access Permissions click on “Edit Limits” button (not all operating systems has this one so skip if yours does not)
  - Add these users: “Everyone”, “ANONYMOUS LOGON”.
  - Select COM Security Tab. Under Launch and Activation Permissions click on “Edit Default” button.
  - Add these users: “ANONYMOUS LOGON”, “INTERACTIVE”.
  - Select COM Security Tab. Under Launch and Activation Permissions click on “Edit Limits” button (not all operating systems has this one so skip if yours does not).
  - Add these users: “ANONYMOUS LOGON”, “INTERACTIVE”.
- Reboot the PC.

## 13 Troubleshooting Hints

This section is provided to help the user identify and correct problems that may arise as a result of incorrectly setting up the OPC90 Server interface. It is provided as a general guide to allow the user to decipher normal and abnormal operation. If this does not help, OPC90 Server can be enabled to post additional error messages to OPC90 Server log files using the DEVICE block DEBUG\_LOG attributes. To change these attributes right mouse click on the Device block and select or deselect Show Receive, Show Send, Show Errors and/or Show Events. Note these flags can be changed in runtime. Remember to save the configuration if these flags are to be saved. Use these DEBUG\_LOG attributes to track down tough problems and discover hard to find configuration errors. Afterwards remember to disable all of its options (except Errors), since leaving them enabled can consume large amounts of disk file space.

After installing the OPC90 Server for the first time and thereafter when it is then started up in Monitor mode or when an OPC client connects to it, the OPC90 Server DEVICE block will automatically begin talking with the Bailey interface. The startup pattern will vary based on the type of Bailey interface being utilized. The first thing you should notice is the Bailey interface serial processing card LEDs begin to sequence. Shortly thereafter you may hear the loop interface termination unit relays click on and off several times as the driver is identifying the Bailey interface type. Just prior to downloading the OPC90 Server block database to the Bailey interface, it will be restarted, and the loop interface termination unit relays will click off isolating it from the communication loop. Next you should observe the Bailey interface serial processing card LEDs sequence at a steady rate as the OPC90 Server block database is being downloaded. Upon completing the database download, the Bailey interface will be commanded on-line, at which time the loop interface termination unit relays will click on and the loop interface card LEDs begin to count loop messages. Thereafter, the Bailey interface serial processing card LEDs will sequence steadily based on the exception report poll interval setup by in the DEVICE block and individual OPC90 Server POLL blocks (when utilized).

If you experience problems with establishing communication between the OPC90 Server and the Bailey interface, if possible, it is a good idea to verify the setup by trying to communicate using the Bailey TXTEWS software. Generally if this software functions OK, you should not experience problems with OPC90 Server.

All Bailey interfaces have a series of four or eight red LEDs on the hardware module that processes serial communication and manages its database. Don't confuse this card with the module that handles the interface with the Bailey communication loop which also has a series of LEDs. An indication that communication with the BAILEY interface is occurring can be determined by looking at the serial processing card LEDs. (Hereafter, these LEDs will be referenced as Bailey interface LEDs.) The Bailey interface LEDs count commands and replies occurring between the OPC90 Server Application Station computer and Bailey interface.

### 13.1 SENTINEL.SYS Causes PC bluescreen on bootup

When OPC90 is started, it immediately attempts to validate its license by communicating with the sentinel.sys driver. When OPC90 is setup to run automatically in an auto logon startup environment, it can sometimes try to communicate with the sentinel .sys driver that has not yet been started by the operating system as a system service. This can result in the infamous PC bluescreen of death.

The solution is to use a program called startme.exe to start OPC90. This program is installed in the C:\Program Files\OPC90 Server directory. Write a batch command file that gets run by the auto logon startup. The batch file should contain the following line:

```
"C:\Program Files\OPC90 Server\Startme.exe" "C:\Program Files\OPC90
Server\OPC90Server.exe" 10000
```

The above command line instructs the STARTME program to start OPC90 in 10000 milliseconds. Note that the start delay time may have to be adjusted to give the operating system more time to start all of the services.

### 13.2 Validating State Of Individual OPC90 Server Blocks

All OPC90 Server blocks have an attribute called MESSAGE. The purpose of this attribute is to help diagnose the operational state of the block. This attribute cannot be connected to from an OPC client but can be displayed in Monitor mode in the bottom status pane when the block is selected. This attribute can take on the following messages:

**OFFLINE:** OPC90 is not currently in monitor mode so the block is not being scanned or OPC90 is unable to communicate with the Bailey interface to receive the block data.

**WAITING ON DEVICE:** The block is waiting for its associated DEVICE block to complete startup of the Bailey interface and make itself available to the other OPC90 Server blocks.

**ESTABLISHING POINT:** The block has requested its associated DEVICE to establish the point in the Bailey interface.

**WAITING FOR DATA FROM BAILEY:** The point has been established in the Bailey interface and is waiting to receive its initial data from the Bailey system.

**ONLINE:** The block has received its first data update from the Bailey system.

**ILLEGAL BLOCK NUMBER:** The OPC90 Server AOL or DOL block has been configured with a block number that has exceeded the DEVICE MAX\_OUTPUTS attribute setting.

**BLOCK ADDRESS ALREADY USED BY ANOTHER BLOCK:** Another OPC90 Server block has been configured for the Bailey address set within this block.

**EXCEEDED BAILEY INTERFACE INDEX CAPACITY:** More OPC90 Server blocks have been configured than can be handled by the Bailey interface.

**EXCEEDED OPC90 Server POINT LICENSE:** More OPC90 Server blocks have been configured than are allowed by the current license.

### **13.3 SCSI Port Not Visible In Device Block Properties Comm Port List Box**

When the PC is first booted up, the SCSI cards scan their buses to determine what devices are present. OPC90 Server scans each SCSI card device list, looking specifically for Bailey interfaces. SCSI port addresses associated with ABB Bailey interfaces are only posted in the Device block properties primary and secondary communication port list boxes if they were found in the SCSI cards device lists.

- 1.) Make sure the ABB Bailey SCSI interface is powered up and not in the “red light” condition. Also verify the ABB Bailey SCSI interface is correctly cabled to one of the PC SCSI cards. This cable connects to the Bailey SCSI interface with a standard 50 pin connector and to the back of the PC SCSI card with a high density 50 pin connector (some cards might use a standard 50 pin connector). ABB sells this cable or it can be obtain from a company such as Black Box ([www.blackbox.com](http://www.blackbox.com)).
- 2.) After reviewing step 1 reboot the PC.
- 3.) If running on Windows 7 OPC90 must be set to XP compatibility mode. This is accomplished by right clicking on the C:\Program Files\OPC90 Server\OPC90Server.exe file and select properties and then XP compatibility mode.

### **13.4 SCSI Communication Not Working on DELL PC**

On some Dell PCs (i.e. model 730) it has been observed the SCSI card BIOS recognizes the ABB IIMCP02 module on the SCSI bus ok but OPC90 or SCSIscan doesn't show the SCSI address associated with that card. This problem can be resolved by making sure the SCSI card driver provided by the manufacturer is properly installed and disabling the embedded SATA driver in the Dell PC BIOS.

### **13.5 SCSI Addresses Different Between Two Shadowed PCs**



The SCSI address of ABB Bailey INICT03A/13A modules must be identical between shadowed OPC90 PCs. The address assigned to SCSI devices has three components which are port number, card number and device on that card. For example the address S2107 indicates port two, card one and device seven. Windows automatically assigns the port number based on when new hardware is added. Therefore, sometimes this port number assignment can end up being different between two PCs based on the order in which new hardware is discovered. If this occurs between shadowed OPC90 PCs use the following procedure to re-align the port address.

Using Regedit go to the key named HKEY\_LOCAL\_MACHINE\HARDWARE\DEVICEMAP\Scsi in the registry. Find the branch that contains the Bailey interface. Click the Scsi Port # at the top of the branch and look for the Driver name.

Go to HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\*drivername* for both the device ahead of you and for the SCSI device you want to be first and look at the TAG value. You need to change the tag value so that the device you want listed first (as S1000) is a lower value than the device that is forcing it to S2000:.

Detailed explanation of what this procedure accomplishes follows as gleaned from the Microsoft web site:

Tag REG\_DWORD Specifies a load order within a given group. The value of Tag specifies a number that is unique within the group of which the service is a member. The related GroupName entry under the Control\GroupOrderList subkey specifies a list of tags, in load order. For example, the following services that are members of the Primary Disk group could have these values: Tag=4 for the Abiosdsk subkey, Tag=2 for Atdisk, Tag=1 for Cpqarray, and Tag=3 for Floppy. The value for Primary Disk under the GroupOrderList subkey will use these Tag values to specify the defined order for loading these services. As another example, each SCSI miniport service has a unique Tag value that is used as an identifier in the SCSI miniport value under the GroupOrderList subkey to define which SCSI adapter to load first.

## 13.6 No Communication

If the Bailey interface LEDs do not sequence, this means the driver is not able to successfully communicate with the N90 interface.

- 1.) Verify the OPC90 Server DEVICE block Primary Port and Secondary Port are associated with the same COM port to which the Bailey interface has been cabled.



- 2.) Verify the port communication settings match those setup within the Bailey interface. Select Edit | Ports and select the correct port for which to verify.
- 3.) Verify the Bailey interface device termination unit/module serial port jumpers are setup correctly.
- 4.) Verify RS232 cable is connected to the correct Bailey termination unit/module connector. Generally this is labeled as the terminal port for the primary channel and printer port for secondary channel when the DEVICE block SCHEME attribute is set to dual channel single interface.
- 5.) Verify the RS232 cable is connected to the correct PC COM port.

### **13.7 Communication Port is reported as Bad or Can't Open**

Use Windows Device Manager to verify the port really exists. Verify there isn't another program running that might have that port open. Make sure the OPC90Server.exe file has its compatibility setting marked as "Run as Administrator". This is accomplished using Microsoft Explorer. Right click on C:\Program Files (x86)\OPC90 Server\OPC90Server.exe and select Properties. Click on the Compatibility tab and make sure "Run As Administrator" is indicated. If not, click it and then click OK. Restart the OPC90 service.

### 13.8 Appears To Be Communicating But No Data is Being Received

If the Bailey interface LEDs sequence at a very steady and periodic rate this means the server is connected to the Bailey interface but the OPC90 Server DEVICE block communication parameters might not be set correctly. These settings are determined by the DEVICE block's Primary and Secondary port properties. Note that these attributes can be changed while the driver is on-line.

- 1.) Verify the baud rate settings match between the Bailey interface and COM port the OPC90 Server DEVICE block driver is using.
- 2.) Verify the parity (usually none) and stop bits (usually 1) match between the Bailey interface and COM port the server is using.
- 3.) Verify the Bailey interface has checksumming enabled.

If the Bailey interface LEDs are not sequencing but the server indicates good communication status (DEVICE PRI\_STATUS or SEC\_STATUS is zero).

- 1.) Verify the COM port associated with the DEVICE is not a modem port.
- 2.) Verify the pin outs on the serial cable or Bailey interface termination dipshunt settings do not have the TX and RX lines pinned together.

### 13.9 Not All Blocks Are Receiving Data

- 1.) Verify the Bailey block addressed by the OPC90 Server block exists and their types match.
- 2.) Verify "BLOCK ADDRESS ALREADY USED BY ANOTHER BLOCK" is not posted in the MESSAGE attribute in any of the OPC90 Server blocks not receiving data. If such a message is found, you have two or more OPC90 Server blocks pointing to the same address within Bailey. Eliminate the duplication.
- 3.) Verify "WAITING FOR DATA FROM BAILEY" is not posted in the MESSAGE attribute in any of the OPC90 Server blocks not receiving data. If such a message is found, it usually indicates the OPC90 block type does not match the ABB Bailey function block type. An example would be to have an OPC90 AIL block trying to receive data from an ABB Bailey function code 222 block. For this problem, the OPC90 block type needs to be HAI.
- 4.) Verify "EXCEEDED OPC90 Server POINT LICENSE" is not posted in the MESSAGE attribute in any of the OPC90 Server blocks not receiving data. If such a message is found, you have exceeded the lite version block capacity. Purchase the non-lite version.

- 3.) Make sure none of the Bailey controllers have addresses less than two.
- 4.) Indices or memory capacity of the Bailey interface has been exceeded. It is unlikely that you will encounter this error unless your application has a very large number of OPC90 Server blocks and the Bailey interface device is a NCIU01 or INCIC01. Enable the Device DEBUG\_LOG error reporting option and search the OPC90 log file messages for such an error.
- 5.) Memory capacity of the Bailey PCU node has been exceeded, check its status using a OPC90 Server MODSTAT block. Note that this error is extremely rare.

### 13.10 Random Blocks Are Receiving Bad Quality Indication

- 1.) Verify the total number of blocks have not exceeded the capacity of the Bailey interface or OPC90 license limit.
- 2.) When this occurs with data values received via DCOM run DCOMCNFG and delete all of the default protocols out of the Default Protocols list except the "Connection-Oriented TCP/IP" protocol.

### 13.11 OSI PI Cannot Read VT\_BOOL Tags Correctly

OPC Foundation defines VT\_BOOL tag values as FALSE = 0, TRUE = -1. The PI OPC client driver doesn't handle that correctly unless you set the PI tag location 2 field (signal conditioning) to a value of 2 for tags of type VT\_BOOL. It can be left at the default value of 0 for all other tag types. Here are the details from the PI OPC client driver manual.

#### Location2 (data-type handling)

Location2 configures handling of data types.

Valid settings for Location2 are as follows:

Value	Description
0	Normal processing; no special handling is used.
1	Read and write value as a string. For digital points, the strings read from the OPC server must match the strings in the digital state set used by the PI point. For integer and real points, the OPC interface requests a string value, and translates it to a number.
2	Read value as a Boolean. Booleans have only two possible values, zero and nonzero. For numeric points, any value but 0 (False) is set to -1 (True). Use this option to correctly convert an OPC server Boolean into the PI Digital State, to prevent the PI point from receiving Bad quality values for a Boolean when it is True.
3	Read value as a four-byte integer. This setting is provided to accommodate servers which cannot send the value as a two-byte integer, which is how Digital points are normally read.
4	Stores the quality of the item rather than the value.
5	Request real points as VT_R8 items (eight-byte real). By default, the PI OPC interface requests real points as VT_R4 items (four-byte real). For float32 points (including all PI2 Real points), values that cannot fit in a 32-bit floating-point number lose precision. This setting is included to support servers that do not translate VT_R8 data to VT_R4 data and to permit the use of float32 points where the benefit of greater precision is not worth the overhead of using float64 points.

### 13.12 Sporadic Data Transfer of Export Blocks to Bailey

The OPC90 AOL, DOL, ODD, OMSDD, ORCM, ORMC, ORMSC and OSTN blocks are used to export data of the named type to Bailey consoles or controllers. Unfortunately, the ABB Bailey CIU interfaces are primarily designed as data importing devices and handle these export blocks at a much lower priority than the import point types. If the OPC90 database contains a large number of import blocks, the CIU can become so busy processing the incoming exception reports that it never gets around to sending the export block data that OPC90 successfully commanded it to send. For example when the OPC client writes to the OPC90 AOL block IN tag, OPC90 transfers that value to the OUT tag after it has been successfully written to the Bailey CIU. If that value is never received within the Bailey system (or received at significant delays), this signifies the ABB Bailey CIU is overwhelmed with processing import exception reports. There are four possible solutions to this ABB Bailey CIU problem.

- 1.) Use combinations of OPC90 and Bailey RMSC and RCM blocks to send analog and digital data.
- 2.) Move the OPC90 export blocks onto a separate ABB Bailey CIU dedicated for these types of blocks.
- 3.) Upgrade the ABB Bailey CIU to a newer model.
- 4.) Determine if ABB Bailey has a newer CIU firmware revision that might correct this problem.



### 13.13 Cannot Export or Control Data Within Bailey

- 1.) Verify that the Bailey interface device supports this type of activity.
- 2.) Verify the allowable ranges for the attribute being written match those configured within Bailey. For example you cannot write an OUT value that exceeds the OUT\_SCALE limits.
- 3.) Verify the Bailey interface does not have monitor mode enabled (very unlikely). Monitor mode is set using an ASCII terminal attached to the termination unit/module printer port and that port selected for the utility mode (consult Bailey interface manual). Normally, monitor mode is disabled when a Bailey interface is received from the Bailey factory. It is unlikely that monitor mode is enabled unless someone had attached an ASCII terminal at one time and was "experimenting" with the available options within the Bailey interface utility menus.
- 4.) Verify the Device is not "locked" by looking at the Device LOCK parameter.

### 13.14 Cannot see all documented block attributes

Occasionally new attributes are added to a block definition that were not defined in an earlier release of OPC90. If you notice a documented block attribute does not appear when the block is selected it could be a new attribute. One such attribute is RED\_TAG. It was added in OPC90 Version 2.1. Configurations saved prior to version 2.1 did not have this attribute defined so therefore it does not automatically appear when the software is upgraded. To make new attributes appear you must export the configuration using the File->export function, request a new configuration using File->new and then import the configuration just exported.

### 13.15 OPC90 Server Will Not Time Sync Bailey

- 1.) Verify that the Bailey interface device is not a NSPM01 or NCIU01. These devices do not allow a computer to send time synchronization commands to Bailey. If the Bailey interface device is one of these types, disable time synchronization.
- 2.) Verify that time synchronization has been enabled. This can be found under the properties of the OPC90 Device  block.
- 3.) For ABB Bailey Network 90 systems, verify that the node map is configured correctly for ALL nodes in the system regardless of whether or not data is being exchanged with any particular node. This can be found under the properties of the OPC90 Device  block.
- 4.) For ABB Bailey Infi 90 systems, time synchronization does not start until after 9 minutes have transpired after OPC90 Server first starts communicating with its interface. This delay is 3 minutes for ABB Bailey Network 90 systems.
- 5.) The PC running OPC90 Server automatically becomes the time synchronization master whenever its clock is changed by two minutes or more.
- 6.) Note that OPC90 Server gives precedence to MCS and OIS nodes as becoming the time synchronization master if its accuracy rating is less than the accuracy rating of those console types. The accuracy rating of the current time sync master can be viewed from the Device block faceplate. The accuracy rating of the OPC90 CIU is set within the Device block properties dialog.

### 13.16 OPC Client Does Not See Any OPC Servers When Browsing Remote PC

A service called OPCEnum must be running on the PC to enable browsing of its OPC servers. Check if this process is running using the Task Manager, process

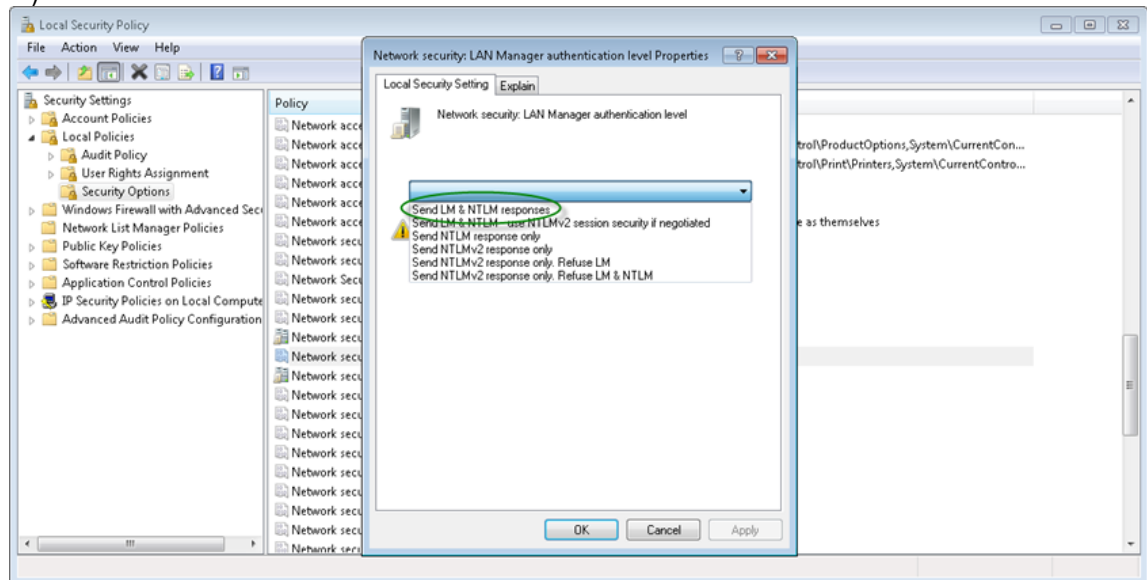
list. If it is running, make sure the Windows “Network access: Let Everyone permissions apply to anonymous users” local security policy is enabled.

The OPC90 install set does not install OPCEnum. It does however include the OPC Foundation Core Components installation MSI file. This file can be found in the C:\Program Files (x86)\OPC90 Server\OPC Foundation directory. Double click on the .MSI file to install these core components which includes the OPCEnum program.

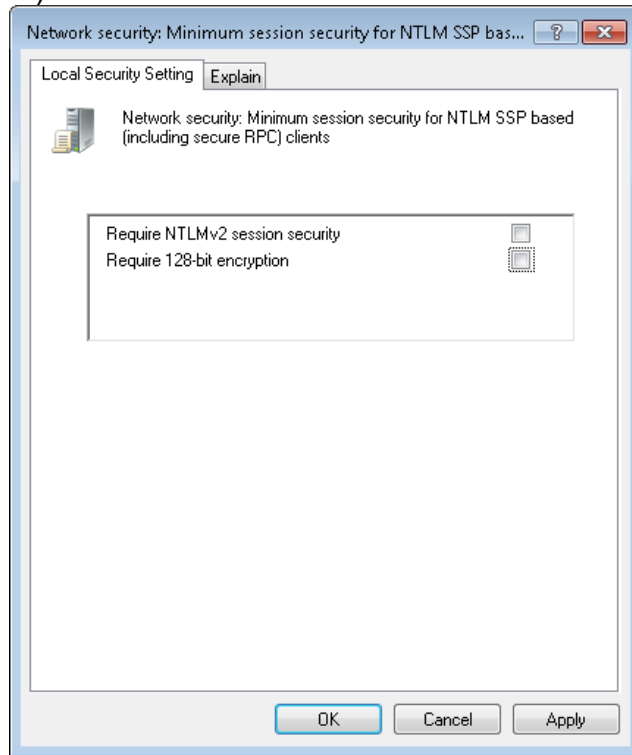
Afterwards, from a command prompt type OPCEnum /Service to install OPCEnum as a service. Depending on the operating system a reboot of the PC might be necessary. The DCOM settings of OPCEnum might also need to be adjusted.

Another possibility is the Windows LAN Manager authentication level isn't setup or it's made too secure. Make the following changes using local security policy setting.

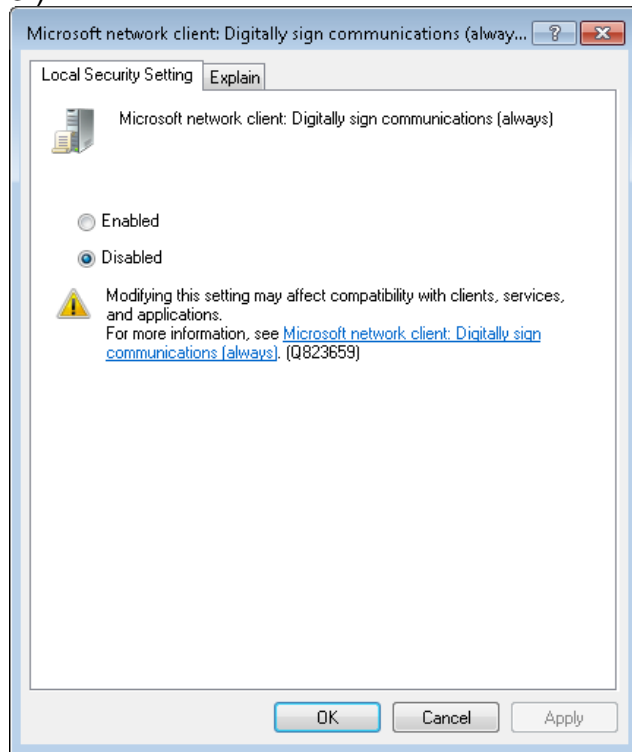
1.)



2.)



3.)



### **13.17 OPC Client Connects But Does Not Receive Data**

Improper DCOM settings can cause multiple instances of OPC90 to run on the local PC. Only the first instance can communicate with the Bailey interface. Verify that a single OPC90 process is displayed in the task manager process list. Note that when a user runs OPC90 manually it may run as a second instance. This is normal. If multiple OPC90 processes still show up in the task manager list after the manually run instance is exited, improper DCOM settings are causing this problem.

### **13.18 Viewing Data in Monitor Mode Does Not Work**

- 1.) Verify the Bailey interface does not require a physical reset. Click on the Device block and look for this message in the program status section (lower right side of the program window).
- 2.) Verify "Data does not exist in first program instance" is not being displayed in the program status section. This message indicates the OPC90 service or embedded instance is not in sync with the second instance. This error is rare but could occur if database changes were made and not saved. Stop and start the OPC90 service.

### **13.19 Writing RMSC Blocks Cause Controller or Node Offline Condition**

A problem exists with ABB Bailey backup controllers or nodes going offline when many writes to RMSC blocks occur. The solution is to limit the number of writes to RMSC blocks to one per second. A common mistake is to use RMSC blocks to exchange values generated by OPC clients and ABB Bailey controllers. This solution is wrong and bypasses the exception report mechanisms that are built into the ABB DCS. The correction implementation is to use OPC90 AOL and DOL blocks linked to ABB Bailey AI/I (FC 121) and DI/I (FC 122) blocks.

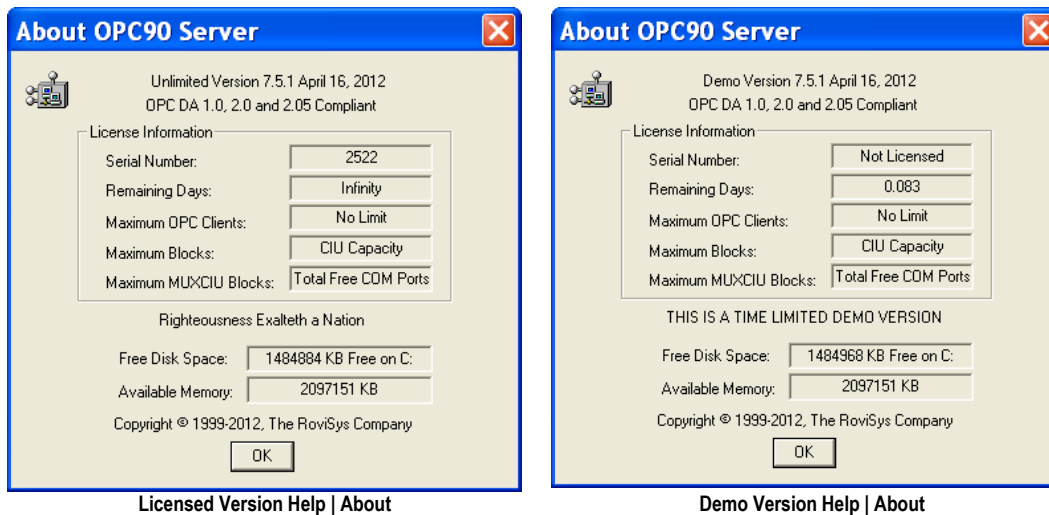
### **13.20 Last Saved Database Not Automatically Restored On Program Start**

- 1.) Verify the last database was saved before exiting OPC90. Run OPC90, select the database and select File->Save. Exit the program and run it again to verify the database is automatically restored.
- 2.) Make sure the user account OPC90 is running in includes the power user group. This is especially critical when running under Microsoft Windows Server.

### **13.21 Data Updates Stop After 1 Hour**



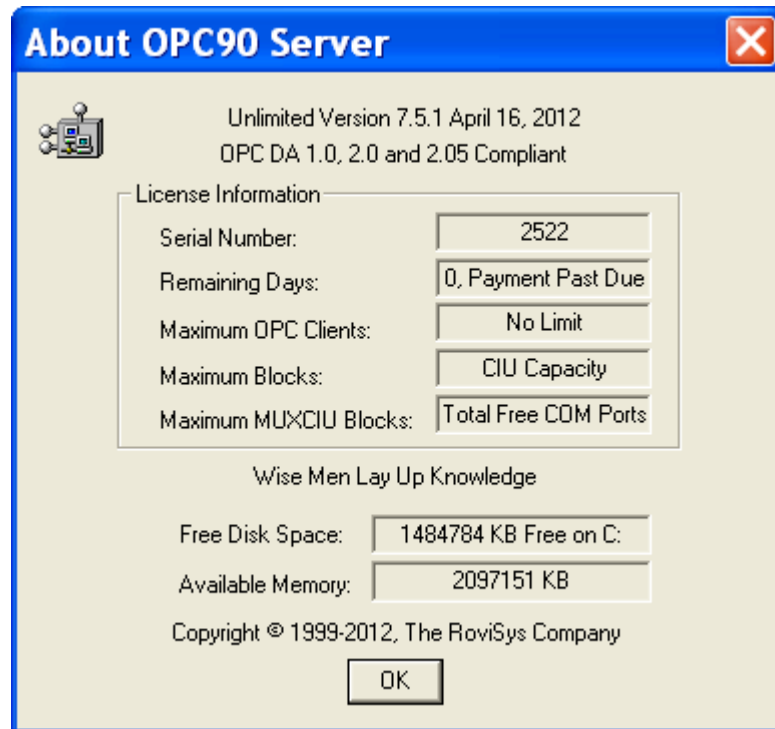
The demo version of OPC90 will only provide data updates for 1 hour each time it is started. If the demo version is being utilized, contact RoviSys to purchase a licensed copy of OPC90. If a licensed copy of OPC90 is installed and the data updates stop after 1 hour, the problem is the hardware protection key (dongle) is not being recognized. Make sure a device block has been configured. OPC90 does not look for its key if no device blocks are configured in the database. If that is not the problem, make sure the dongle is properly installed. Select the OPC90 program menu option Help | About and verify it reports the dongle serial number in the first line of the about dialog. If the dongle is not being recognized or the demo version is being used, that line will only state the program version number and release date. If the demo version was being utilized prior to purchasing a licensed copy, it must be uninstalled and the licensed copy installed.



Note that if the dongle was not properly installed, OPC90 must be restarted after the dongle is installed. If the dongle has been properly installed but OPC90 still is not recognizing it, the Sentinel Protection Installer program must be re-run. The OPC90 setup program normally runs this setup but on some systems it has been observed to occasionally fail. This program can be found in the C:\Program Files\OPC90 Server\SuperPro directory. Stop OPC90 and run the Sentinel Protection Installer program. Restart OPC90 and verify it now correctly reports the dongle serial number. If not, try rebooting the PC and checking it again. If that doesn't work contact RoviSys regarding replacement of the dongle.

### 13.22 Data Updates Stop After 8 Hours

An OPC90 license is being used that has not been paid for. Some OPC90 hardware keys are shipped with a licensed number of days the server is allowed to run. Failure to pay for the server within that amount of time causes the license to expire. Expired licenses will only communicate with the ABB Bailey system for eight hours each time OPC90 is restarted. Check the Help | About menu option to verify this has happened. When the serial number is displayed and the remaining days indicate zero, the license is expired.



If the software has been purchased directly from RoviSys, please ask your company's accounts payable department payment to issue payment. If the software was purchased from a reseller, please contact them about why they have not paid for the software. Once payment is received by RoviSys, the license day limit will be removed. Until that occurs, restart OPC90 every 8 hours. Note that the hardware key does not have to be returned to RoviSys. The license update is handled via email. A license update file is emailed. This file is used to immediately update the hardware key to remove the day limit.

### 13.23 OPC90 Running as a Service Randomly Stops or Crashes

RoviSys has taken every effort to develop a "bug" free OPC server. Nevertheless sometimes unforeseen problems exist caused by loading, random hardware glitches or other factors such as misbehaving OPC client software. These types of problems are usually very difficult to duplicate and isolate. Verify the latest version of OPC90 is installed. Make sure the Windows Dr. Watson program is active on the system. Email the Dr. Watson log to RoviSys for analysis.

### 13.24 Can't Run as a Service Under Windows

To run OPC90 as a service under Windows it must be set to run as an administrator. This is accomplished by right clicking on the C:\Program Files\OPC90 Server\OPC90Server.exe or C:\Program Files\OPC90 Server (x86)\OPC90Server.exe file and select properties. Under the compatibility tab enable "Run as Administrator".

### **13.25 BLK and SPEC Blocks Do Not Report Correct Specification Names**

Sometimes when running OPC90 under a 64 bit Windows operating system the BLK and SPEC blocks may not report the correct specification names. Instead the names are reported as “Not defined in registry entry F1\_130” or “Not defined in registry F131\_255”. This can usually be corrected by logging into an administrator account and then from Explorer double click on the file called “OPC90\_INI\_ADD\_MAPPING.REG” found in the “C:\Program Files (x86) \OPC90 Server” directory. Answer yes to the inquiry to allow addition to the registry. After the registry update is completed, restart the computer.

### **13.26 CIU Red Lights with LEDs 1, 4, 5 and 6 On**

When the CIU red lights with LEDs 1, 4, 5 and 6 on this means it was commanded to go online but discovered another node at the same address as itself so it cannot continue to operate. This condition does not indicate a faulty CIU. Recheck the CIU switch settings to make sure they are setup for a unique address within the ABB Bailey system.

Note when using redundant OPC90 installations that will be sourcing values to the ABB Bailey system (using OPC90 AOL, DOL, ODD, OMSDD, ORCM, ORMC, ORMSC and OSTN blocks), same address CIUs may be desirable and intentionally setup this way. Same address CIUs are supported by OPC90 and utilizes the configuration shadowing feature to share critical CIU information and operational states between the redundant OPC90 servers. Therefore, configuration shadowing must be enabled on both OPC90 servers and reported as “Good” in each OPC90 program status (bottom of program window). If configuration shadowing reports “Bad” or “Off” on either OPC90, both will attempt to command their CIU online, resulting in this red light condition. After configuration shadowing has been properly setup and working correctly, reset the CIU in the red light state to recover its use.

### **13.27 OPC90 Will Only Run in Demo Mode Or Reports Demo Expired**

The OPC90 downloaded from the RoviSys, OPC Foundation and OPC Training Institute websites is the demo only version. By design that version can only run in demo mode. Verify the demo version is not installed by looking at the OPC90 Help | About dialog. If it states “THIS IS A TIME LIMITED DEMO VERSION” purchase a licensed version from RoviSys.

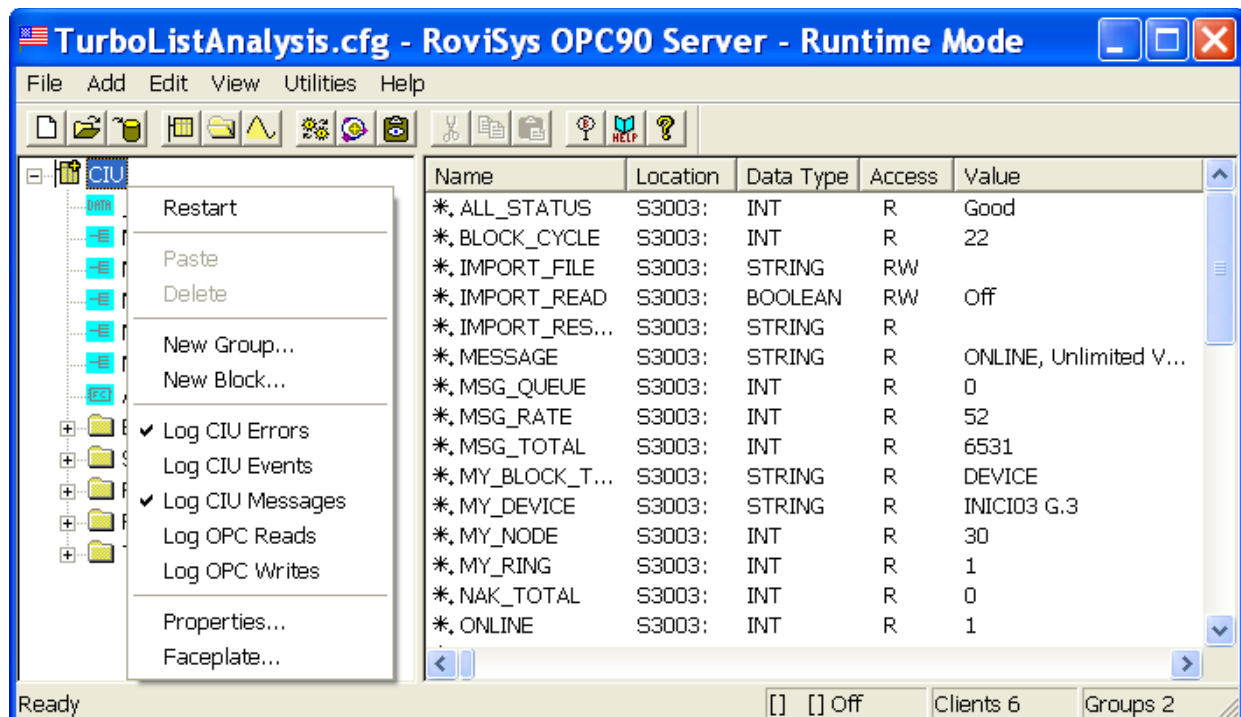
If a licensed version has been installed but it is still running as demo there are several things to check.

- 1.) Make sure the USB license key is plugged into the PC.
- 2.) Run License Manager desktop short cut and select Report button. Verify it reports the license serial number.
- 3.) Run Windows Service Manager and verify the RoviSys License Service is running. If it is not running, start it and then restart OPC90 (restart OPC90 service).

- 4.) Stop RoviSys License Service and then set the compatibility of C:\Program Files\RoviSys\LicenseManager\RVSService.exe or C:\Program Files (x86)\RoviSys\LicenseManager\RVSService.exe to Run as an administrator. Start the service back up and restart OPC90 (restart OPC90 service).
- 5.) Re-run the C:\Program Files\RoviSys\LicenseManager\Sentinel Protection Installer 7.x.x.exe or C:\Program Files (x86)\RoviSys\LicenseManager\Sentinel Protection Installer 7.x.x.exe file and allow it to re-install or fix itself. Restart the PC.

### 13.28 Forcing a CIU Restart

Although forcing a CIU restart is rarely needed OPC90 supports such action. Let's say a CIU stops updating a set of points or it just doesn't appear to be running normal. The OPC90 program window can be used to request the CIU to be restarted. This is accomplished by right clicking on the DEVICE block (called CIU in this example) and selecting Restart. The user will be prompted to make sure a restart is really wanted before it is done. Restarting a CIU will temporarily cause the tag value statuses OPC90 is providing to OPC clients set to bad. Data flow values for those tags will also temporarily be paused while the restart occurs.



Note that if OPC90 is not currently in run time mode of operation, the Restart option will be grayed out.

### **13.29 Can't Establish Communication with IET800**

The Device block message tag reports "Primary interface basic security problem, see help troubleshooting". The IET800 requires an ABB Basic Security license to be installed and the module setup correctly to support Ethernet or serial communication. Review "Using OPC90 With IET800" section of this document and verify correct setup.

Another possibility is ABB has supplied a modified IET800 that has the high speed serial channels permanently disabled. Switch to using the Ethernet port or find a different IET800 that doesn't have these channels disabled.

### **13.30 IET800 Ethernet Communication is Really Slow**

IET800 Ethernet communication requires definition of a registry key called TcpAckFrequency . Review "OPC90 Ethernet CommunicationSetup" section for details where to add this key.

### **13.31 Blocks Reporting Waiting for Specifications**

There are three things that happen when blocks are starting up. The first is the block is established in the CIU database. The .MESSAGE tag will report "Establishing Point" until this occurs. Afterwards the block will report "Waiting for data from Bailey". After receiving data it will report "ONLINE, Waiting for Specifications". A block could be in this state for minutes before the CIU acquires the block specifications. OPC90 monitors blocks for this state and if they remain too long will request the specifications are regenerated. If after 5 minutes the block remains in this state it could be an issue with the PCU node or the block type doesn't match the type in the ABB controller. Check for a match and if OK, often forcing a failover of the redundant PCU node NPM/NIS pair will resolve the issue.

*Corporate Headquarters:*

The RoviSys Company  
1455 Danner Drive  
Aurora, Ohio 44202  
(330) 562-8600  
[www.rovisys.com](http://www.rovisys.com)