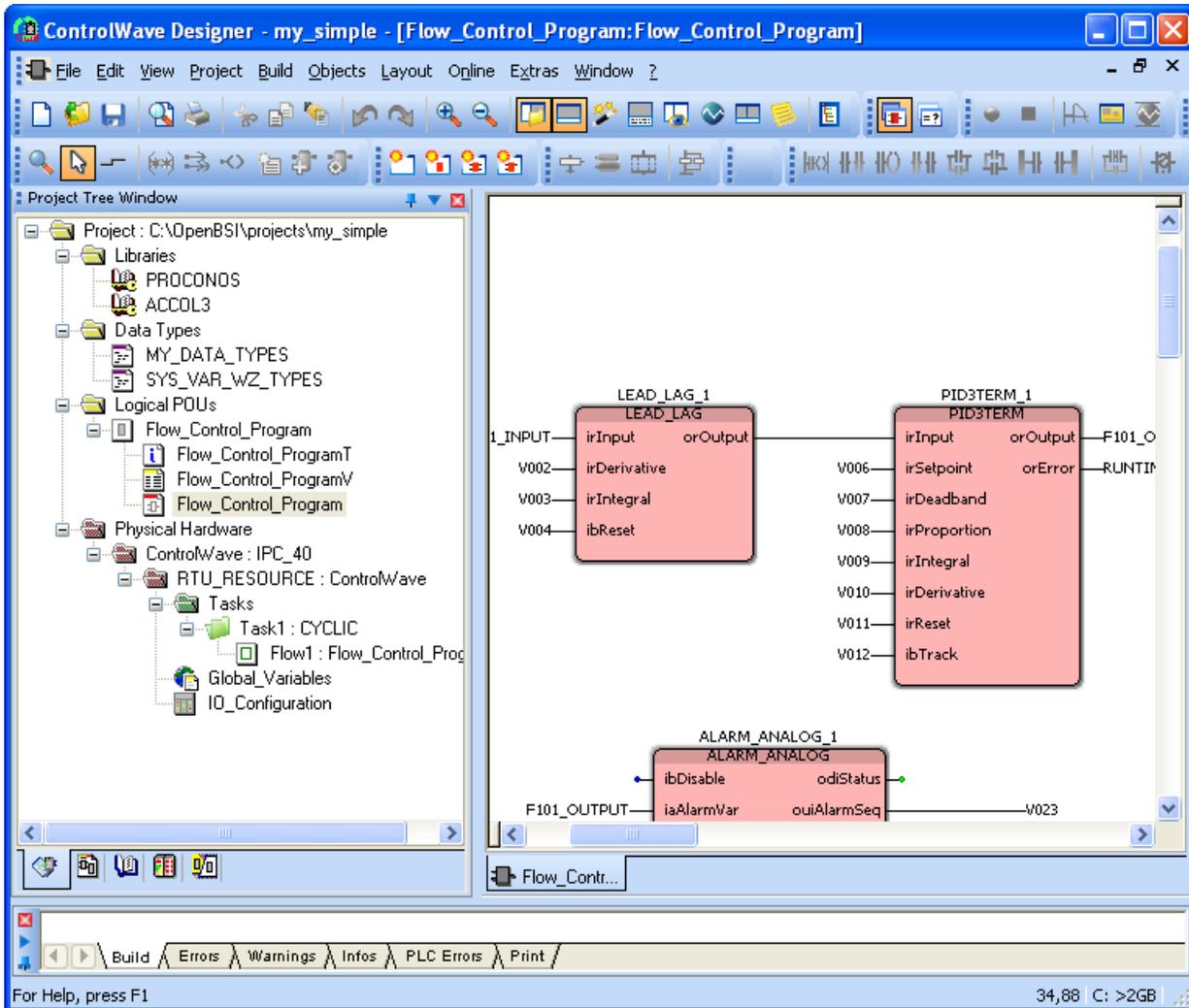


ControlWave™ Designer Programmer's Handbook



Application Safety Considerations

Protecting Operating Processes

A failure of this application – for whatever reason -- may leave an operating process without appropriate protection and could result in possible damage to property or injury to persons. To protect against this, you should review the need for additional backup equipment or provide alternate means of protection (such as alarm devices, output limiting, fail-safe valves, relief valves, emergency shutoffs, emergency switches, etc.).

System Training

A well-trained workforce is critical to the success of your operation. Knowing how to correctly install, configure, program, calibrate, and troubleshoot your Emerson equipment provides your engineers and technicians with the skills and confidence to optimize your investment. Energy and Transportation Solutions offers a variety of ways for your personnel to acquire essential system expertise. Our full-time professional instructors can conduct classroom training at several of our corporate offices, at your site, or even at your regional Emerson office. You can also receive the same quality training via our live, interactive Emerson Virtual Classroom and save on travel costs. For our complete schedule and further information, contact the Energy and Transportation Solutions Training Department at 800-338-8158 or email us at education@emerson.com.

Contents

Introduction	1
ControlWave Product Line	1
Manuals You Should Read <u>Before</u> You Read This One	1
What is Covered in this Manual?	1
What Related Documentation is Available?	2
ACCOL III Function Block Library	5
Alarm Configuration	9
What are Alarms?	9
Where can I get detailed information about these function blocks?	11
Configuring an Analog Alarm.....	12
Using the ALARM_ANALOG function block	13
Configuring a Logical Alarm.....	17
Using the ALARM_LOGICAL_ON function block	17
Configuring a Change of State Alarm	19
Application Licensing	23
Granting a License for a Controller to Run a Standard Application	23
Removing an Application License from a Controller:	24
Viewing a History of Dongle Issue / Remove Operations:	25
Application Parameters	27
Archive Configuration	29
What Are Archive Files?	29
Archive configuration involves four basic steps:	30
What can be done with the data from the Archive File(s)?	31
Step 1. Define Archive Files(s) in the Flash Configuration Utility	31
Step 2. In Your ControlWave Designer Project, Identify the variables you want to archive in the Archive List	36
Step 3. Create an Output List for Accessing the Most Recent Archive Record (OPTIONAL)	37
Step 4. Configure the ARCHIVE Function Block.....	38
Array Configuration	41
Audit Configuration	43
Step 1. Set parameters in the Flash Configuration Utility	43

Step 2. In ControlWave Designer, identify Variables for which you want to maintain Audit Logging46

Step 3. Configure an AUDIT Function Block..... 47

BSAP Addressing and Networks **49**

What is BSAP?..... 49

Adding A ControlWave to an OpenBSI BSAP Network in the RTU Wizard 51

Setting the BSAP Local Address and EBSAP Group 52

What is Client/Server Communication?..... 53

BSAP - Underlying Technical Details (For ADVANCED USERS) 53

BSAP Master Port **55**

Configuring A BSAP Master Port..... 56

BSAP Slave Port **61**

Configuring a BSAP Slave Port..... 61

Communication Ports **67**

Compiling **95**

Conditional Logic **97**

DataView **101**

Before you begin: 101

Calling up ControlWave Data in DataView 102

Debugging – An Overview **105**

Starting Debug Mode 105

Using the Watch Window 105

Using the Cross-Reference Window 107

On-line Editing with Patch POU 108

Using the Force/Overwrite Options..... 109

Setting a Breakpoint 110

Exiting Debug Mode 112

Downloading **113**

Two Methods Available for Downloading 113

Downloading from within ControlWave Designer 114

Downloading Your ControlWave Project from Within ControlWave Designer 117

Downloading using the OpenBSI ControlWave Downloader..... 119

Starting the ControlWave Downloader 121

Using the ControlWave Downloader 122

Creating Download Scripts for Batch Downloading of ControlWave Controllers.....	123
Expanded BSAP (EBSAP) Communications	127
Expanded BSAP – The Concept	128
General Requirements for Expanded BSAP (EBSAP):.....	129
Creating an EBSAP Master.....	130
OpenBSI Workstation is EBSAP Master	130
ControlWave-series Controller is the EBSAP Master	131
Configuring the Control and Status Arrays	132
Defining the Virtual Nodes	141
Defining the EBSAP Slave Nodes	142
Example 1 – OpenBSI Workstation is EBSAP Master to 1000 ControlWave controllers	143
Example 2 – ControlWave Controller is EBSAP Master to 300 ControlWave EBSAP Slaves	146
Flash Configuration Utility – An Overview	149
Starting the Flash Configuration Utility	149
Flash File Access	155
Viewing a List of Files in the Flash File Area:	156
Uploading a File from the ControlWave to Your OpenBSI Workstation:	156
Copying a File from the OpenBSI Workstation to Your ControlWave:	157
Deleting a File from the ControlWave User Flash Files Area:	157
Refreshing the List of Files:	158
Function Blocks – Creating	159
Function Block Parameter Name Prefixes	165
Historical Data	167
What is Historical Data Used For?	167
What types of Historical Data can be saved in the ControlWave Controller?.....	167
How is Audit Trail and Archive Data Retrieved from the ControlWave Controller?	168
I/O Configurator	169
Tables of Board Types	174
Analog Boards	179
Digital Boards	183
High Speed Counter (HSC) Boards	185
Remote I/O Status Board	186
System Controller Board.....	186
CWM_RTU Board.....	187
Notes About Ethernet I/O Boards.....	187
HART Interface Board (CWM_HIB)	193

I/O Mapping **197**

Common Device Map	197
Local I/O - ControlWave	198
Ethernet I/O.....	206
ControlWave I/O Expansion Rack Boards.....	218
Local I/O – ControlWave MICRO-series.....	229
Local I/O – ControlWave GFC-CL and ControlWave XFC	242
Local I/O – ControlWave Express and ControlWave GFC	248
Local I/O – ControlWave CW10, CW30, CW35	256
I/O – ControlWave CW_31.....	261
ControlWave MICRO I/O Expansion Rack	269

I/O Simulator **283**

What is the I/O Simulator?	283
Starting the I/O Simulator	283
Analog Boards	287
Digital Boards	288
Counter Boards.....	288
Viewing the Board Configuration Status	289
Configuring a Pin	289
Viewing Simulated Alarms	290
Shutting Down the I/O Simulator	290
Troubleshooting Tip	290

IP Addressing and Networks **293**

What is the Format of IP Addresses?	293
Adding a ControlWave to an IP Network with the RTU Wizard	298
Setting up IP Ports in the Flash Configuration Utility.....	299
Recommended Ranges for IP Addresses	299

IP Parameters **301**

IP Ports - Ethernet **307**

IP Ports – PPP **309**

IP Routes **311**

Libraries **315**

Memory Usage **319**

Some Background - What is Memory?.....	319
What is Downloading?.....	319

Types of Memory in the ControlWave Process Automation Controller (CW PAC).....	320
What happens in the event of a power failure or the power switch is turned off?	323
What happens in the event of a watchdog condition?	323
What happens on restart after a power failure or watchdog?	324
Variations when using ControlWave MICRO/EFM.....	327
Variations when using ControlWave GFC/GFC-CL, XFC, Corrector, Express or ExpressPAC.....	328
Variations when using ControlWave_10/_30/_35 (CW_10, CW_30, CW_35)	329
Memory Allocation Issues	330
Determining POU Size at Compilation Time.....	330
Resolving “Not Enough Memory” Messages	330
Modbus Configuration	335
Configuring Your ControlWave Controller as a Modbus Master Device.....	335
Configuring Your ControlWave Controller as a Modbus Slave or Enron Modbus Slave Device	339
Modbus Ports	341
Configuring Modbus Ports.....	341
Reset ControlWave Utility	343
Security	345
Other Security-Related Issues	351
Security Protocols (CHAP and PAP)	355
Security Protocols (CHAP and PAP) Used on PPP Links.....	355
Challenge Handshaking Authentication Protocol (CHAP).....	355
Password Authentication Protocol (PAP)	358
Security Protocols (DNP3 SAv5)	361
SAv5 Implementation Considerations.....	361
Locking Down the ControlWave Device and Application	362
System Tasks (Warm/Cold Starts)	365
System Variables	367
Using the System Variable Wizard	367
System Variable Mapping Charts	371
Static Memory Area:	397
Using the System Variable Viewer.....	400
Variable Extension Wizard	403
Before You Begin	403

Starting the Variable Extension Wizard	403
Using the Variable Extension Wizard	404
Marking a Variable for Report by Exception (RBE) Collection	407
Configuring a Variable as an Alarm.....	408
Creating / Editing a List	410
Setting initial values for Manual or Alarm Inhibit/Enable Flags	412
Assigning Units Text (Analog Variables ONLY).....	413
Assigning ON/OFF Text (BOOL Variables ONLY)	414
Creating Descriptive Text for the Variable	415
Saving the Initialization Files and Exiting the Wizard.....	416
Format of Initialization Files	416
Troubleshooting Tips.....	421

Variables and Data Types **423**

Global Variables Vs. Local Variables	423
Variable Addressing.....	424
System Variables	425
Data Types	425
Notes about STRING variables	426

Variable Naming Conventions **427**

Versions and Compatibility **429**

Virtual Ports **445**

VSAT Slave Port – Configuration **447**

Configuring Flash Parameters	447
Configuring System Variables	448
VSAT Master Ports	450

Web Pages – Notes About Using **451**

Other Notes About Using Web Pages.....	452
Calling Up Web_BSI Pages	453
Creating Your Own Web Pages to Use with the ControlWave	455

Appendix A: Troubleshooting Tips **457**

Using the Debug Information Tool.....	462
Other Debugging Tools (BTCSPy, DLM Monitor).....	466

Appendix B: ControlWave Designer Compatibility Issues **467**

Bringing an Older ControlWave Project into a Newer Version of ControlWave Designer	467
Warning - I/O Configurator and Multiple Copies of ControlWave Designer	468

Introduction

ControlWave Product Line

Unless otherwise noted, the information in this manual applies to any controller in the ControlWave product line, including:

- ControlWave Process Automation Controller
- ControlWave Low Power (LP) Controller
- ControlWave Redundant Controller
- ControlWave MICRO Controller
- ControlWave Electronic Flow Meter (EFM)
- ControlWave Gas Flow Computer (GFC)
- ControlWave Gas Flow Computer Plus (GFC Plus)
- ControlWave Explosion-Proof Gas Flow Computer (XFC)
- ControlWave Express / Express Process Automation Controller
- ControlWave Corrector

Manuals You Should Read Before You Read This One

Before you read this document, we strongly recommend you read, and try out, the example presented in the Getting Started with ControlWave Designer Manual (part number D301416X012). It is designed to explain various concepts to first-time ControlWave users.

In addition, please review the quick setup guide for your particular controller (see next page for the proper document) which contains notes about how to initially set up your ControlWave Controller, and how to configure certain parameters for first-time use.

The ControlWave Designer Programmer's Handbook (which you are reading right now) builds on the information contained in these other documents, so it is essential that you are familiar with the material included in them.

What is Covered in this Manual?

The *ControlWave Designer Programmer's Handbook* is intended to provide you the information you need to get the most out of your ControlWave Designer software. It includes:

- Examples of how to configure various sub-systems of ControlWave software, which are commonly required in process control applications, such as: alarming, and historical data collection.
- Instructions for how to create your own function blocks, and how to create a library of them, so they can be re-used in other projects.
- Notes about how to use OpenBSI, and BSAP, or IP, to include your ControlWave in a network.
- Explanations about how ControlWave memory works, and how I/O points can be configured.
- Discussions of communication options, as well as how to download your ControlWave project into the ControlWave controller.

What Related Documentation is Available?

For information on this...	Please consult this...
<ul style="list-style-type: none"> ▪ ControlWave Designer 	<i>Getting Started with ControlWave Designer</i> (part D301416X012)
<ul style="list-style-type: none"> ▪ IEC 61131 terminology and languages ▪ Projects, Project Tree, POU's, Tasks, Resources ▪ ACCOL3 library ▪ PROCONOS library 	Online help in ControlWave Designer, accessible through the question mark [?] menu item.
<ul style="list-style-type: none"> ▪ Flash Configuration 	Chapter 5 of the <i>OpenBSI Utilities Manual</i> (part number D301414X012) contains full details on flash configuration.
<ul style="list-style-type: none"> ▪ Web Pages ▪ ActiveX controls used with ControlWave 	<i>Web_BSI Manual</i> (part number D301418X012).
<ul style="list-style-type: none"> ▪ Converting ACCOL II source files (*.ACC) into ControlWave Projects 	<i>ACCOL Translator User's Guide</i> (part number D301417X012)
<ul style="list-style-type: none"> ▪ ControlWave Process Automation Controller 	<i>ControlWave Quick Setup Guide</i> (part number D301415X012)
<ul style="list-style-type: none"> ▪ ControlWave Low Power (LP) Controller 	<i>ControlWave LP Quick Setup Guide</i> (part number D301422X012)
<ul style="list-style-type: none"> ▪ ControlWave I/O Expansion Rack 	<i>ControlWave I/O Expansion Rack Quick Setup Guide</i> (part number D301423X012)
<ul style="list-style-type: none"> ▪ ControlWave Redundant Controller 	<i>ControlWave Redundancy Setup Guide</i> (part number D301424X012).
<ul style="list-style-type: none"> ▪ ControlWave MICRO Controller 	<i>ControlWave Micro Quick Setup Guide</i> (part number D301425X012)
<ul style="list-style-type: none"> ▪ ControlWave Electronic Flow Meter (EFM) 	<i>ControlWave Electronic Flow Meter (EFM) Instruction Manual</i> (part number D301383X012)
<ul style="list-style-type: none"> ▪ ControlWave Gas Flow Computer (GFC) 	<i>ControlWave Gas Flow Computer (GFC) Instruction Manual</i> (part number D301387X012)

For information on this...	Please consult this...
<ul style="list-style-type: none"> ▪ ControlWave Explosion-Proof Gas Flow Computer (XFC) 	<i>ControlWave Explosion-Proof Gas Flow Computer (XFC) Instruction Manual</i> (part number D301396X012)
<ul style="list-style-type: none"> ▪ ControlWave Express 	<i>ControlWave Express RTU Instruction Manual</i> (part number D301386X012)
<ul style="list-style-type: none"> ▪ ControlWave Express PAC 	<i>ControlWave Express PAC Instruction Manual</i> (part number D301384X012)
<ul style="list-style-type: none"> ▪ ControlWave Corrector 	<i>ControlWave Corrector Instruction Manual</i> (part number D301382X012)
<ul style="list-style-type: none"> ▪ ControlWave Gas Flow Computer Plus 	<i>ControlWave Gas Flow Computer Plus Instruction Manual</i> (part number D301389X012)
<ul style="list-style-type: none"> ▪ ControlWave Ethernet I/O 	<i>ControlWave Ethernet I/O Instruction Manual</i> (part number D301395X012)
<ul style="list-style-type: none"> ▪ ControlWave Industrial Ethernet Real Time Switches 	<i>ControlWave Industrial Ethernet Real-time Switches Instruction Manual</i> (part number D301390X012)

Note:

Because the existing ControlWave install-base includes customers with older hardware and software, for support purposes this manual includes references to older un-supported operating systems and devices.

ACCOL III Function Block Library

The ACCOL III Function Block Library is a set of functions and function blocks created by Emerson and included with ControlWave Designer. ACCOL III function blocks are designed to provide ControlWave Designer with the capabilities of ACCOL II modules used in our previous ACCOL II language. This library also includes several newer functions which were not available in ACCOL II.

Note

Other function block libraries are available for liquids calculations, and NIST23 calculations. Contact Emerson Energy and Transportation Solutions for more information.

For instructions on how to use any of these function blocks, please consult the online help in ControlWave Designer.

Function block or Function	Minimum Firmware Revision Required	Description
AGA3	2.00.00	Computes natural gas volume flow rate through an orifice plate in thousands of cubic feet per hour (MSCFH) according to American Gas Association (AGA) Report #3 1985 Edition.
AGA3DENS	4.10.00	Computes mass and volume flow rate for fluids (gases or liquids) in lbs/hour and cubic ft per hour, for orifice plates, with flange taps ONLY, according to the method explained in the American Gas Association (AGA) Report #3 of August, 1992, 3rd Edition (Part 1 and Part 4).
AGA3I	1.00.00	Computes natural gas volume flow rate according to the Factors Method in AGA Report #3.
AGA3SELECT	5.50.00	Combines the functions of the AGA3I and AGA3TERM function blocks into a single function block.
AGA3TERM	2.00.00	Computes natural gas volume flow rate through an orifice plate in thousands of cubic feet per hour (MSCFH) according to AGA Report #3 1985 Edition. Allows factor substitution and display.
AGA5	2.00.00	Performs AGA-5 calculations for conversion of computed gas volume to energy equivalents.
AGA7	2.00.00	Performs AGA-7 calculations for base volume rate.
AGA8_DET2017	6.00.00	Computes Base compressibility, Flowing compressibility and Supercompressibility for natural gas mixtures, according to the Detail Characterization Method in the 2017 AGA Report 8 Part 1.
AGA8DETAIL	2.00.00	Computes Base compressibility, Flowing compressibility and Supercompressibility for natural gas mixtures, according to the Detail Characterization Method in AGA Report 8.
AGA8GROS	1.00.00	Computations for natural gas mixtures according to the Gross Characterization method in AGA Report 8.
AGA8_GRS2017	6.00.00	Computations for natural gas mixtures according to the Gross Characterization method in the 2017 AGA Report 8 Part 1.

Function block or Function	Minimum Firmware Revision Required	Description
AGA8_PART2	6.00.00	Performs calculations specified by 2017 AGA Report 8 Part 2, Thermodynamic Properties of Natural Gas and Related Gases, GERG–2008 Equation of State.
AGA10	4.50.00	Computations for natural gas mixtures according to AGA Report Number 10.
ALARM	4.70.00	Used to batch process alarms defined in the Variable Extension Wizard.
ALARM_ANALOG	1.00.00	Monitors a process variable and sends a notification message to a remote computer when the variable's value exceeds user defined limits.
ALARM_LOGICAL_ON	1.00.00	Monitors a Boolean process variable and sends a notification message to a remote computer when the variable is TRUE.
ALARM_LOGICAL_OFF	1.00.00	Monitors a Boolean process variable and sends a notification message to a remote computer when the variable is FALSE.
ALARM_STATE	1.00.00	Monitors a Boolean process variable and sends a notification message to a remote computer when the variable changes state.
ANOUT	1.00.00	Converts and scales signals for hardware analog output.
ARCHIVE	1.00.00	Provides historical storage of signal values.
ARRAY_ANA_GET	4.50.00	Function. Returns the REAL value of the specified array element.
ARRAY_ANA_SET	4.50.00	Function. Writes a REAL value to the specified array element.
ARRAY_LOG_GET	4.50.00	Function. Returns the BOOL value of the specified array element.
ARRAY_LOG_SET	4.50.00	Function. Writes a BOOL value to the specified array element.
AUDIT	1.00.00	Provides historical storage of alarms and events.
AUDIT_SELECTED	5.40.00	Allows you to log events for value changes of individual variables or log customized events. The customized events can include standard value change information, alternate values, or NOTE events. You can use the AUDIT_SELECTED FB independently of the AUDIT FB.
AUTOADJUST	2.00.00	Performs adjusted volume and self check calculations for an Invensys Auto-adjust Turbine Meter.
AVERAGER	1.00.00	Computes the time-average and integral.
BTI	4.70.00	Allows CW_10, CW_30, and CW_35 controllers with the BBTI board to collect data from the Bristol Teletrans™ Model 3508 Transmitter.
CALC_DENSITY	June 22, 2009 library or newer.	Calculates the mass density of a gas using the real gas relative density (specific gravity)
CLIENT	2.00.00	Communicates with other ControlWave controllers that have implemented the Server Function Block.
COMMAND	1.00.00	Pulse Delayed Output.
COMPARATOR	1.00.00	Analog signal comparison.
CRC	2.00.00	Calculates CRC of data stored in an array.
CUSTOM	1.00.00	Custom Communications Interface.
DACCUMULATOR	1.00.00	Performs Double Precision Arithmetic.
DB_LOAD	4.00.00	Loads variable information (LISTS) from a text file.
DEMUX	1.00.00	Copies a signal value into a list of signals.
DIAL_CTRL	4.00.00	Establishes a dial-up connection on a given port, or interface to a

Function block or Function	Minimum Firmware Revision Required	Description
		modem.
DIFFERENTIATOR	1.00.00	Differentiates an analog signal.
DISPLAY	4.20.00	Provides support for the keypad display.
ENCODE	1.00.00	Set/Read System Time, Julian /Wall Time Conversions.
EVP	4.50.00	Calculates the equilibrium vapor pressure for a liquid.
FIELDBUS	5.10.00	Interfaces with Foundation Fieldbus devices via a bridge server.
FILE_CLOSE	2.20.00	Function – End access to a flash file.
FILE_DELETE	2.20.00	Function – Remove a file from flash.
FILE_DIR	2.20.00	Function Block – Get a listing of files. Retrieve file attributes.
FILE_OPEN	2.20.00	Function – Start access to an existing flash file or create a new one.
FILE_READ	2.20.00	Function – Read binary data from file.
FILE_READ_STR	2.20.00	Function – Read a line from a file and load into a string.
FILE_WRITE	2.20.00	Function – Send a buffer of binary data to a flash file.
FILE_WRITE_STR	2.20.00	Function – Send a formatted string to a flash file.
FPV	2.00.00	Computes Super Compressibility Factor (FPV) of a gas per AGA Report NX-19.
FUNCTION	1.00.00	Table lookup and interpolation with arrays.
GENERIC_SERIAL	2.00.00	Generic Serial communications serves as a means to buffer user defined data through a serial port.
GPA8173	4.50.00	Converts the mass of natural gas liquids to equivalent liquid volumes at base conditions.
GSV	4.50.00	Converts the gross standard volume for a liquid.
HART	5.00.00	Interface to HART field devices via serial port or I/O board.
HILOLIMITER	1.00.00	Compares a signal against a high and low limit.
HILOSELECT	1.00.00	Finds highest and lowest REAL values in a signal list.
HSCOUNT	1.00.00	High Speed Counter.
HWSTI	4.80.00	Honeywell Smart Transmitter Interface
IEC62591	5.50.00	Allows a ControlWave Micro controller with an IEC62591 Interface module to communicate to IEC62591 wireless devices.
INTEGRATOR	1.00.00	Computes an integral approximation.
ISO5167	2.00.00	Calculates flow rate for Orifice plates, Nozzles, Venturi tubes, and Venturi-nozzle Primary Devices per ISO 5167-1980 (E), 1980 edition
LEAD_LAG	1.00.00	Adds a controlled delay effect.
LICENSE	4.90	Determines application licenses for the RTU.
LIQUID_DENSITY	4.50.00	Calculates the density of liquid at flowing conditions.
LISTxxx	1.00.00	Define/Expand a list.
LIST_ELE_NAME	3.00.00	Returns the variable name for a given list element.
MUX	1.00.00	Extracts the value of a signal from a list of signals
PDO	1.00.00	Pulse Duration Output.
PID3TERM	1.00.00	3 Mode PID Control.
PORTATTRIB	4.40.00	Allows the user to set the port characteristics online (except for the MODE).
PORT_CONTROL	4.20.00	Communications port manual control.

Function block or Function	Minimum Firmware Revision Required	Description
R_INT	1.00.00	Function to Truncate to Integer.
R_RND	1.00.00	Function to Round to nearest Integer.
RBE	4.40.00	Supports Report by Exception
RBE_DISABLE	5.60.00	Disables selected RBE variables to prevent RBE reporting from them.
REDUN_SWITCH	2.00.00	Function to perform programmed fail-over to Redundant Standby.
REG_ARRAY	1.00.00	Register Arrays – For HMI Access.
SCHEDULER	3.00.00	Equalizes the elapsed running time of a number of external devices.
SEQUENCER	1.00.00	Provides sequential output control.
SERVER	2.00.00	Communicates with other ControlWave controllers that have implemented the Client Function Block.
STEPPER	1.00.00	Sequence up to 255 Boolean Outputs.
STORAGE	04.40	Stores and retrieve historical data.
TCHECK	4.70.00	Provides status checking and data processing for a 3508 Teletrans Transmitter.
TOT_TRND	1.00.00	Computes totals from input and slope of input.
USERS_ACTIVE	5.20	Returns information on all currently signed-in users.
USERS_DEFINED	5.20	Allows encrypted access to the security configuration of the ControlWave.
V_ATTRIB_GET	04.50	This function returns the value of the specified attribute of a variable.
V_ATTRIB_SET	04.50	This function sets the value of the specified attribute of a variable.
VAR_ATTRIB_GET	2.10.00	Retrieves the value of an attribute for a variable.
VAR_ATTRIB_SET	2.10.00	Sets the value of an attribute for a variable.
VAR_CI_PROC	2.10.00	Performs the CI (Control Inhibit) processing for the given variable.
VAR_FETCH	3.00.00	This function block fetches complete information about a variable given its name.
VAR_SEARCH	3.00.00	This function searches the PDD for variables, both with a given index, and by name.
VIRT_PORT	2.20.00	This function block creates a virtual port by defining a connection to a terminal server.
VLIMIT	1.00.00	Limit an input's rate of change.
VMUX	3.00.00	Extracts the value of a signal from a list of signals and ramps the output to match the input.
WATCHDOG	5.60	Activates a watchdog output based on user-defined criteria.
XMTR	4.70.00	Provides read/write access to the memory of a TeleTrans Transmitter, or other compatible device.

Alarm Configuration

What are Alarms?

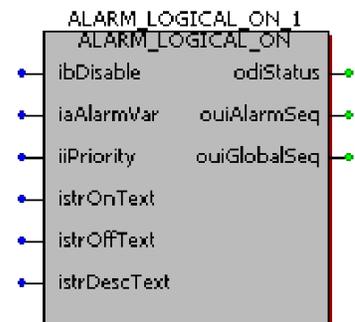
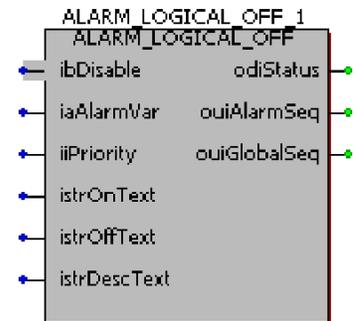
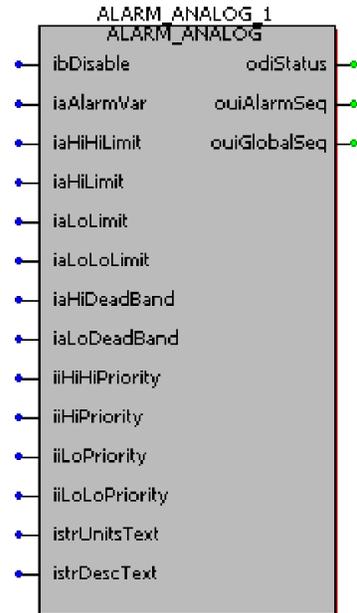
Alarms are messages generated by the controller when one or more process variables changes, and that change violates some pre-defined limit or state.

There are three types of alarms:

Analog Alarms– These alarms are generated when an analog variable (type REAL, INT, etc.) exceeds a pre-defined **alarm limit**. A **return-to-normal** message is generated when the variable returns to within the pre-defined limit. **Deadbands** can be established around the alarm limits so that variables can fluctuate slightly near the alarm limit without constantly going into and out of an alarm state, and thereby flooding the system with repetitive alarms. Analog alarms are configured using the ALARM_ANALOG function block.

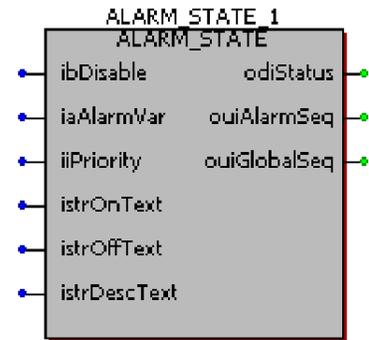
Logical Alarms – These alarms are generated when a variable of type BOOL enters its 'in-alarm' state. A return-to-normal message is generated when the variable returns to its opposite (non-alarm) state. The user chooses which state is the 'in alarm' state by the choice of alarm function block. If its 'in-alarm' state occurs when the BOOL variable becomes TRUE, then the ALARM_LOGICAL_ON function block should be used. If the variable's 'in-alarm' state occurs when the BOOL variable becomes FALSE, then the ALARM_LOGICAL_OFF function block should be used.

Note: The ALARM_LOGICAL_ON and ALARM_LOGICAL_OFF function blocks are identical *except* that the ALARM_LOGICAL_ON function block generates an alarm when the associated process variable (iaAlarmVar) is TRUE, whereas the ALARM_LOGICAL_OFF function block generates an alarm when the associated process variable (iaAlarmVar) is FALSE.



Change-of-State Alarms – These alarms are generated whenever a variable of type BOOL changes state. In this case, there is no such thing as a return-to-normal condition. Change-of-State alarms are configured using the ALARM_STATE function block.

All three types of alarms have various inputs and outputs associated with them. These inputs and outputs are configured within the alarm function block. The following information is common to all three types of alarms:



Alarm Variable This is the actual process variable which is to be monitored by the alarm function block. Analog (REAL, INT, etc.) can only be monitored via the ALARM_ANALOG function block. Boolean variables can be monitored by either the ALARM_LOGICAL_ON, ALARM_LOGICAL_OFF, or ALARM_STATE function block(s).

Alarm Priority The alarm priority is basically a number which indicates the importance of the alarm. There are four priorities supported in the system:

Priority	Value	Meaning
Event	0	Event alarms are used to indicate normal, everyday occurrences.
Operator Guide	1	Operator guide alarms are used to indicate everyday occurrences which are slightly more important than events.
Non-Critical	2	Non-critical alarms are used to indicate problems, which, while not serious enough to cause damage to a plant or process, require corrective action.
Critical	3	Critical alarms are used to indicate dangerous problems that require immediate attention and/or corrective action.

The choice of which alarm priorities are to be assigned to a particular alarm variable is entirely at the discretion of the user.

Descriptive Text Up to 64 characters of descriptive text may be stored along with any alarm message. This text may be changed dynamically by control logic.

Disable Alarm processing for a particular process variable can be turned OFF. This prevents any alarm messages for that process variable from being generated. This might be used in the case of system maintenance or troubleshooting.

Status Every alarm function block maintains error and status information. Improper configuration is indicated by negative status values, and will prevent execution of the alarm function block. Positive values indicate the variable is in an alarm state. A value of '0' indicates there are no

configuration errors, and the variable is NOT in an alarm state. For a full description of status values, see the on-line help files.

Alarm Sequence #	An alarm sequence number is assigned to each alarm message to uniquely identify it within the Alarm System. Alarm sequence numbers range from 0 to 65,535 and are shared among all alarm function blocks in the alarm system to maintain proper ordering of messages.
Global Sequence #	A global sequence number is assigned to each audit record, archive record, or alarm message to uniquely identify it within the ControlWave controller. Global sequence numbers range from 0 to 65,535 and are shared by all of these sub-systems to maintain proper ordering of messages.

The remaining inputs/outputs configured within the Alarm function block vary depending upon the type of alarm function block being used.

Note

Some parameters in the alarm function block may not be required, depending upon your specific application. In addition, if you do NOT intend to change the value of a particular input value, and its parameter only supports a single data type (i.e., its parameter name does NOT have an "ia" or "iany" prefix), you can enter it as a constant, instead of assigning a variable name.

Alternatively, the OpenBSI Alarm Router can be used to view alarms, and offers an interface to custom alarm applications. See the *Open BSI Utilities Manual* (part number D301414X012) for information on Alarm Router.

Where can I get detailed information about these function blocks?

On-line help is provided within ControlWave Designer for every ACCOL3 function block. To access this, you can right click on the ACCOL3 library icon, and choose **"Help on ACCOL 3 Library..."** from the pop-up menu.

Important

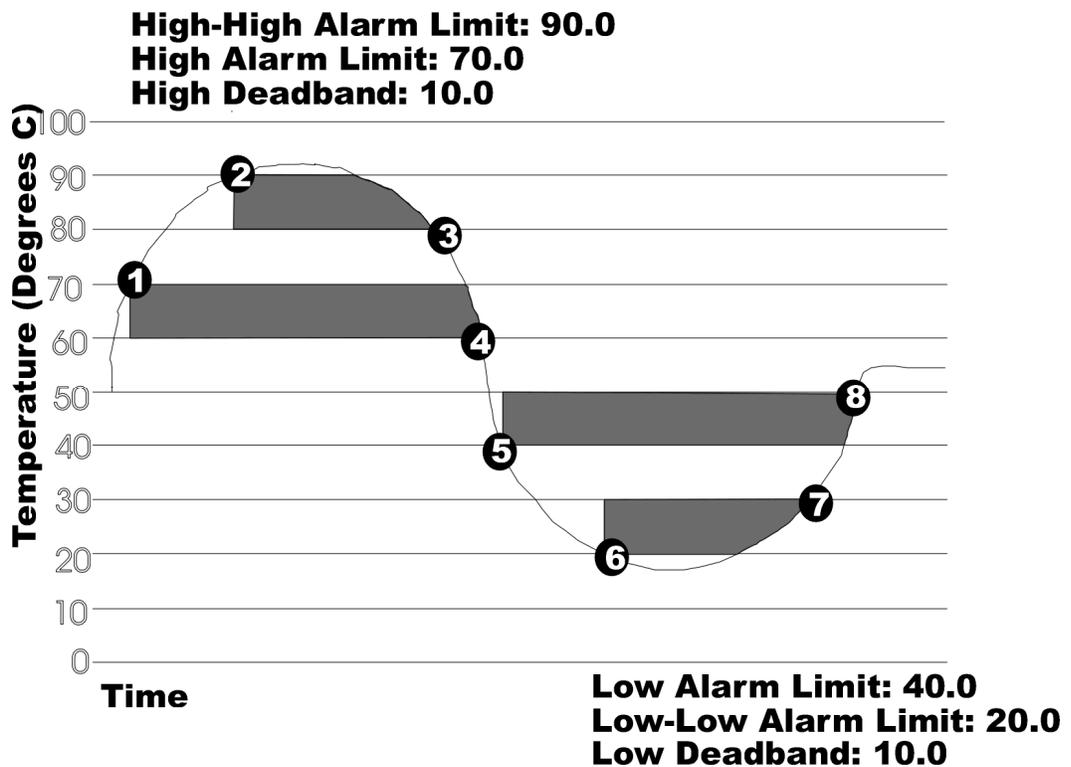
This section describes the standard method for alarm configuration. Beginning with OpenBSI Version 5.4, an alternate way to create alarms is to use the Variable Extension Wizard and the ALARM function block for batch processing. The Variable Extension Wizard configures the alarms in the controller, but they do NOT appear within your project. For information on this method, see the Variable Extension Wizard section, later in this manual.

Configuring an Analog Alarm

For this example, let's say we have a tank full of water which must be maintained at a certain temperature range. A temperature transmitter is mounted on the tank to measure the current temperature of the water, and it has been decided that the water temperature should be kept between 40.0°C and 70.0° Celsius. Any temperature reading outside of that range indicates an alarm condition.

The figure, below, shows a plot of the value of the variable measuring Celsius temperature in the tank, as it fluctuates over time. Four alarm limits and two deadbands have been defined. Starting from the left of the graph, the value of the variable increases until it reaches 70.0° C, the high alarm limit (see Item 1). At this point a high alarm message is generated, and the variable is considered to be in a 'high alarm' state.

The value of the variable continues to increase. When it passes the high-high alarm limit of 90.0° C a 'high-high' alarm message is generated (see Item 2). At this point, the variable is considered to be in a high-high alarm state.



The value of the variable then starts to decrease. Although the value passes below 90.0°C, it is still considered to be in a 'high-high' alarm state because there is a 10.0° high deadband in effect (deadbands are shown as shaded areas on the graph.) When the variable value falls lower than 80.0° C point (90.0° C high alarm limit minus the high deadband of 10.0° C) the variable is no longer in a 'high-high' alarm state (See Item 3). It is still however in a 'high' alarm state.

As the value of the variable decreases below 70.0° C, it remains in a 'high' alarm state until its value falls below 60.0° C (70.0° C alarm limit, minus a 10.0° C high deadband). (See Item 4). At this point, the variable is in its normal range, and a 'return-to-normal' alarm message is sent.

Then, however, the value of the variable continues to drop. When it reaches 40.0° C, a 'Low Alarm' message is generated (See Item 5).

The variable remains in a 'Low Alarm' state until the variable value drops to 20.0° C. (See Item 6). This causes a 'Low Low Alarm' message to be generated.

The variable remains in a 'Low-Low Alarm' state until the variable rises above 30.0° C, (20.0° C low-low alarm limit plus low deadband of 10.0° C). (See Item 7). The variable is still in a 'Low Alarm' state, however.

Once the variable rises above 50.0° C (40.0° C low alarm limit + low deadband of 10.0° C), it has left the low-alarm state, and a 'return to normal' alarm message is sent (See Item 8).

As long as the variable remains in the normal range (between 40.0 and 70.0° C), no more alarm messages will be generated.

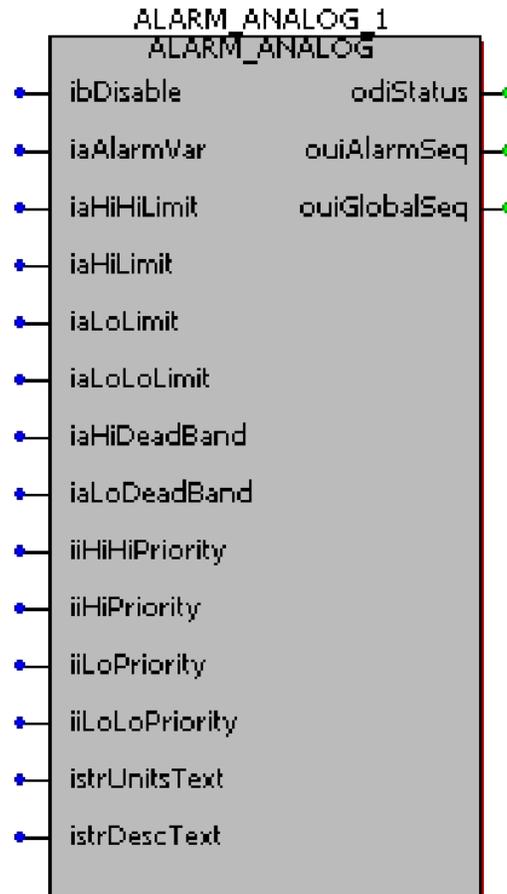
Using the ALARM_ANALOG function block

Where you insert your ALARM_ANALOG function block depends upon the overall construction of your project:

The ALARM_ANALOG function block can only detect an alarm condition *when it is executed*. Therefore, when constructing your project, you should think about how and when you want to execute the function block.

- You might choose to place the ALARM_ANALOG function block in the *same* POU which holds primary control logic for the process variable. This is always required if the variable being monitored is not global. Typically, the ALARM_ANALOG function block would be placed at the end of the POU. If you require timestamps to reflect the exact moment the alarm condition occurred, you could place the ALARM_ANALOG function block immediately following the logic which manipulates the process variable.
- Alternatively, you could place the ALARM_ANALOG function block in a separate POU, which could even be executed at a different task interval. This approach may be convenient if you want to organize your project such that all alarm function blocks are in the same place.

Step 1. Following the considerations discussed above, insert an ALARM_ANALOG function block into your program. A separate ALARM_ANALOG function block is required for every analog variable you want to configure as an alarm.



Step 2. Based on our example of monitoring water temperature in a tank, assign meaningful variable names, and where applicable, initial values to each of the parameters of the ALARM_ANALOG function block.

The actual process variable being monitored for alarm conditions must be assigned to the iaAlarmVar parameter. In this case we will call it WATER_TEMP since that is the process I/O variable from the temperature transmitter.

Now we need to assign alarm limits and deadbands based on our previous discussion of the proper range for the water temperature.

Important

You must use the *same* variable type for alarm limits and alarm deadbands as you use for your alarm variable. For example, if the alarm variable is defined as type REAL, then every alarm limit and alarm deadband for that ALARM_ANALOG function block must also be defined as type REAL. Type mismatches of any kind will prevent the ALARM_ANALOG function block from working and will generate errors on the odiStatus parameter.

Setting Alarm Limits (iaLoLimit, iaLoLoLimit, iaHiLimit, iaHiHiLimit)

We want to generate an alarm when the temperature goes out of the range 40.0° Celsius to 70.0° Celsius, so 40.0 will be our low alarm limit (iaLoLimit parameter) and 70.0 will be our high alarm limit (iaHiLimit parameter). We can also assign Low-Low and High-High alarm limits to generate additional alarm messages if the variable goes much higher or much lower than the Low and High limits. These are on the iaHiHiLimit and iaLoLoLimit parameters, respectively. NOTE: You do not need to use all four limits. Only one alarm limit is required to configure an analog alarm.

Setting Deadbands (iaHiDeadBand, iaLoDeadBand)

We strongly recommend that you define deadbands around your alarm limits. Two deadbands are supported, iaHiDeadBand is applied to the high and high-high alarm limits, and iaLoDeadBand is applied to the low and low-low alarm limits. Deadbands are ranges above a low limit, or below a high limit, in which a return-to-normal message will NOT be sent, even though a process variable has returned inside the range defined by the alarm limits. This is to prevent the system from being flooded with alarm messages if a process variable is fluctuating around the alarm limit. Without a deadband defined, every time the process variable enters or leaves the normal range, a return-to-normal or alarm message would be generated, thereby flooding the system with repetitive alarms, even though the process variable has changed very little. For this example, we have chosen a deadband of 10.0 for both the low and high deadbands.

Setting Alarm Priorities (iiLoPriority, iiLoLoPriority, iiHiPriority, iiHiHiPriority)

Alarm priorities indicate the severity of the alarm condition triggered by passing one of the pre-defined alarm limits. For this example, passing either the low or high alarm limits is considered NON-CRITICAL, and passing either the low-low or high-high alarm limits is considered CRITICAL. The alarm priority has no effect on the operation of the alarm system, and is defined strictly at the user's discretion. Priorities are displayed as part of the alarm message.

Units Text, Descriptive Text (istrUnitsText, istrDescText)

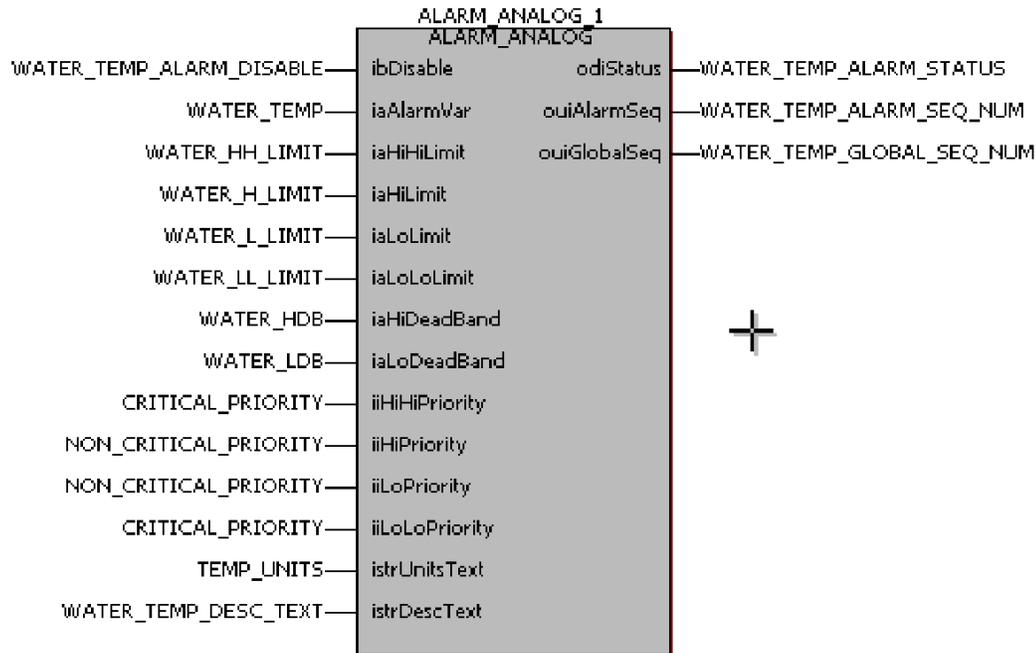
These parameters are strictly for the convenience of the user. They specify engineering units for the alarm variable, and descriptive text for the alarm condition.

The table, below, summarizes the values for the various parameters used in this example:

Parameter Name	Suggested Variable Name	Variable Type	Value	Notes
ibDisable	WATER_TEMP_ALARM_DISABLE	BOOL	FALSE	For disabling/ enabling alarm processing.
iaAlarmVar	WATER_TEMP	REAL, SINT, INT, DINT, USINT, UINT, or UDINT	None	The actual process variable measuring temperature in the tank.
iaHiHiLimit	WATER_HH_LIMIT	REAL, SINT, USINT, INT,	90.0	High-high alarm limit

Parameter Name	Suggested Variable Name	Variable Type	Value	Notes
		UINT, DINT, or UDINT		
iaHiLimit	WATER_H_LIMIT	REAL, SINT, USINT, INT, UNIT, DINT or UDINT	70.0	High alarm limit
iaLoLimit	WATER_L_LIMIT	REAL, SINT, USINT, INT, UNIT, DINT or UDINT	40.0	Low alarm limit
iaLoLoLimit	WATER_LL_LIMIT	REAL, SINT, USINT, INT, UNIT, DINT or UDINT	20.0	Low low alarm limit
iaHiDeadBand	WATER_HDB	REAL, SINT, USINT, INT, UNIT, DINT or UDINT	10.0	High alarm deadband
iaLoDeadBand	WATER_LDB	REAL, SINT, USINT, INT, UNIT, DINT or UDINT	10.0	Low alarm deadband
iiPriority	CRITICAL_PRIORITY	INT	3	High-High Priority
iiHiPriority	NON_CRITICAL_PRIORITY	INT	2	High Priority
iiLoPriority	NON_CRITICAL_PRIORITY	INT	2	Low Priority
iiLoLoPriority	CRITICAL_PRIORITY	INT	3	Low-Low Priority
istrUnitsText	TEMP_UNITS	STRING	'DEG_C'	Engineering units (up to 6 characters)
istrDescText	WATER_TEMP_DESC_TEXT	STRING	'WATER TEMPERATURE'	Descriptive text (up to 64 characters)
odiStatus	WATER_TEMP_ALARM_STATUS	DINT	None	Status of the execution of this function block.
ouiAlarmSeq	WATER_TEMP_ALARM_SEQ_NUM	UINT	None	Alarm sequence number.
ouiGlobalSeq	WATER_TEMP_GLOBAL_SEQ_NUM	UINT	None	Global sequence number.

The configured alarm block appears, below:



Configuring a Logical Alarm

For this example, let's say we have an electrical switch which turns ON in the event of a compressor power failure. When the switch turns ON, we want to generate an alarm. When the power is restored, the switch turns OFF and a return-to-normal message will be generated.

Because the alarm condition occurs when the switch turns ON, we must use an ALARM_LOGICAL_ON function block. (If we wanted to generate an alarm when the switch turned OFF, we would have used an ALARM_LOGICAL_OFF function block.)

Using the ALARM_LOGICAL_ON function block

Where you insert your ALARM_LOGICAL_ON function block depends upon the overall construction of your project:

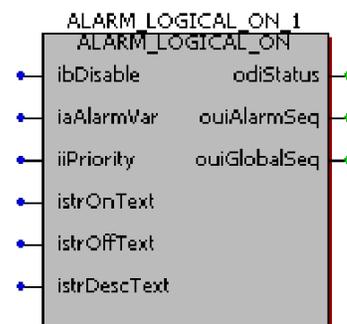
- The ALARM_LOGICAL_ON function block can only detect an alarm condition *when it is executed*. Therefore, when constructing your project, you should think about how and when you want to execute the function block.
- You might choose to place the ALARM_LOGICAL_ON function block in the *same* POU which holds primary control logic for the process variable. This is always required if the variable being monitored is not global. Typically, the ALARM_LOGICAL_ON function block would be placed at the end of the POU. If you require timestamps to reflect the

exact moment the alarm condition occurred, you could place the ALARM_LOGICAL_ON function block immediately following the logic which manipulates the process variable.

- Alternatively, you could place the ALARM_LOGICAL_ON function block in a separate POU, which could even be executed at a different task interval. This approach may be convenient if you want to organize your project such that all alarm function blocks are in the same place.

Step 1. Following the considerations discussed above, insert an ALARM_LOGICAL_ON function block into your program. A separate ALARM_LOGICAL_ON (or ALARM_LOGICAL_OFF) function block is required for every variable you want to configure as a logical alarm.

Step 2. Based on our example of detecting a change of state of a pump, assign meaningful variable names, and where applicable, initial values to each of the parameters of the ALARM_LOGICAL_ON function block.



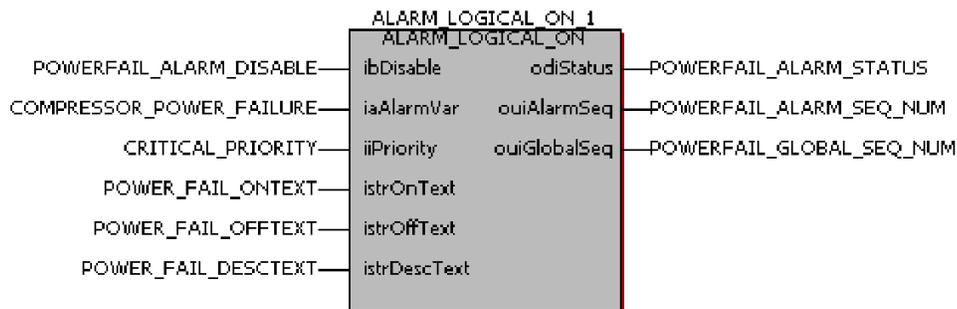
The actual process variable being monitored for alarm conditions must be assigned to the iaAlarmVar parameter. In this case we will call it COMPRESSOR_POWER_FAILURE since that is the process I/O variable from the switch.

The table, below, summarizes the values for the various parameters used in this example.

Parameter Name	Suggested Variable Name	Variable Type	Value	Notes
ibDisable	POWERFAIL_ALARM_DISABLE	BOOL	FALSE	For disabling/ enabling alarm processing.
iaAlarmVar	COMPRESSOR_POWER_FAILURE	BOOL	None	The ON/OFF status of the switch used to indicate power failure of the compressor. NOTE: Alarms are only generated when iaAlarmVar is ON.
iiPriority	CRITICAL_PRIORITY	INT	3	Priority of this alarm condition.
istrOnText	POWER_FAIL_ONTEXT	STRING	'FAILED'	ON message text (up to 6 characters)
istrOffText	POWER_FAIL_OFFTEXT	STRING	'NORMAL'	OFF message text (up to 6 characters)
istrDescText	POWERFAIL_DESC_TEXT	STRING	'WATER TEMPERAT URE'	Descriptive text (up to 64 characters)

Parameter Name	Suggested Variable Name	Variable Type	Value	Notes
odiStatus	POWERFAIL_ALARM_STATUS	DINT	None	Status of the execution of this function block.
ouiAlarmSeq	POWERFAIL_ALARM_SEQ_NUM	UINT	None	Alarm sequence number
ouiGlobalSeq	POWERFAIL_GLOBAL_SEQ_NUM	UINT	None	Global sequence number

The configured alarm block appears, below:



Configuring a Change of State Alarm

For this example, let's say we have a pump which is critical to the operation of a water plant. Whenever it starts or stops, we want to generate an alarm message. NOTE: In change-of-state alarms, there is NO return-to-normal state; every change from ON-to-OFF or OFF-to-ON generates an alarm message.

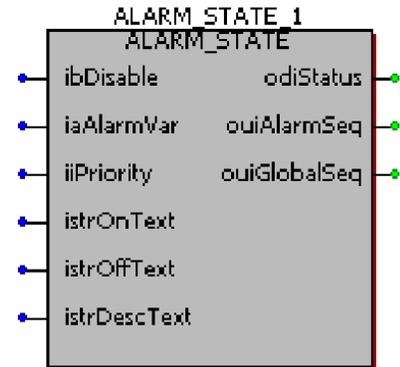
Using the ALARM_STATE function block

Where you insert your ALARM_STATE function block depends upon the overall construction of your project:

- The ALARM_STATE function block can only detect an alarm condition *when it is executed*. Therefore, when constructing your project, you should think about how and when you want to execute the function block.
- You might choose to place the ALARM_STATE function block in the *same* POU which holds primary control logic for the process variable. This is always required if the variable being monitored is not global. Typically, the ALARM_STATE function block would be placed at the end of the POU. If you require timestamps to reflect the exact moment the alarm condition occurred, you could place the ALARM_STATE function block immediately following the logic which manipulates the process variable.
- Alternatively, you could place the ALARM_STATE function block in a separate POU, which could even be executed at a different task interval. This approach may be

convenient if you want to organize your project such that all alarm function blocks are in the same place.

Step 1. Following the considerations discussed above, insert an ALARM_STATE function block into your program. A separate ALARM_STATE function block is required for every variable you want to configure as a change-of-state alarm.



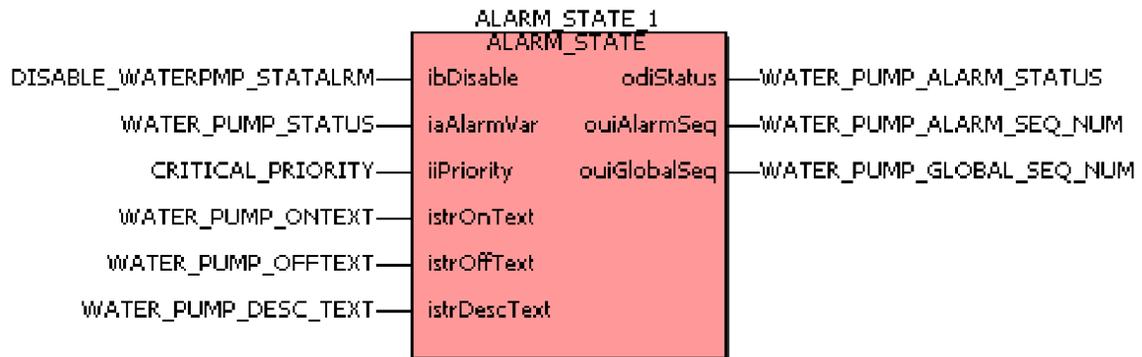
Step 2. Based on our example of detecting a change of state of a water pump, assign meaningful variable names, and where applicable, initial values to each of the parameters of the ALARM_STATE function block.

The actual process variable being monitored for alarm conditions must be assigned to the iaAlarmVar parameter. In this case we will call it WATER_PUMP_STATUS.

The table, below, summarizes the values for the various parameters used in this example:

Parameter Name	Suggested Variable Name	Variable Type	Value	Notes
ibDisable	DISABLE_WATERPMP_STATARM	BOOL	FALSE	For disabling/ enabling alarm processing.
iaAlarmVar	WATER_PUMP_STATUS	BOOL	None	The ON/OFF status of the water pump.
iiPriority	CRITICAL_PRIORITY	INT	3	Priority of this alarm condition.
istrOnText	WATER_PUMP_ONTEXT	STRING	'ACTIVE'	ON message text (up to 6 characters)
istrOffText	WATER_PUMP_OFFTEXT	STRING	'IDLE'	OFF message text (up to 6 characters)
istrDescText	WATER_PUMP_DESC_TEXT	STRING	'WATER PUMP STATE CHANGE'	Descriptive text (up to 64 characters)
odiStatus	WATER_PUMP_ALARM_STATUS	DINT	None	Status of the execution of this function block.
ouiAlarmSeq	WATER_PUMP_ALARM_SEQ_NUM	UINT	None	Alarm sequence number
ouiGlobalSeq	WATER_PUMP_GLOBAL_SEQ_NUM	UINT	None	Global sequence number

The configured alarm block appears, below:



Application Licensing

Certain standard applications sold by Emerson Energy and Transportation Solutions are licensed to prevent unauthorized duplication. When you purchase the application(s), you will receive a license key (also known as a USB dongle), that will allow you to manage which controllers will be license recipients for the standard applications.

Note:

Only ControlWave-series controllers with firmware 04.90 or newer support application licensing.

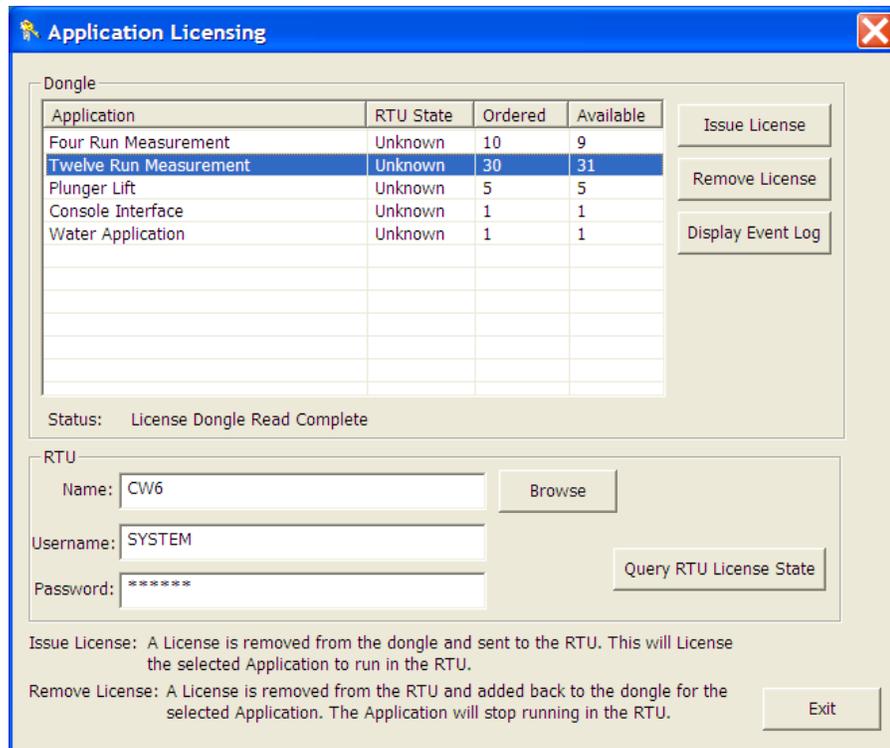
Granting a License for a Controller to Run a Standard Application

1. Install the dongle in your PC's USB port, and allow it to be recognized.
2. Start the Application Licensing Tool, using the following sequence:

Start → Programs → OpenBSI Tools → ControlWave Tools → Application Licensing

3. The Application Licensing Tool will read the dongle, and display the following information:

Application	A list of all the standard application licenses originally issued via this dongle.
RTU State	The license status of a particular controller for a particular standard application. If the controller has not been queried yet, this will display 'Unknown'.
Ordered	The total number of licenses included on this dongle, for a given application, when it left the factory.
Available	The total number of licenses, for a given application, which have not yet been issued to controller(s).



4. Select the controller to which you want to issue the license, using the **[Browse]** button. (If communication is via NetView, the controller must exist in the current NETDEF file; otherwise select the locally connected 'RTU'.)
5. Enter a valid username / password combination for that controller, and click on the **[Query RTU License State]** button.
6. The application window will be updated to show which applications have been licensed for this controller. Select the application you want to issue to the controller. There must be at least 1 or more listed in the **"Available"** field.
7. Click on **[Issue License]**. The license will be granted to the controller, and the **"Available"** count will be decremented by 1.
8. Repeat this process for any other controllers you want to issue licenses to. When finished, click on **[Exit]**.

Removing an Application License from a Controller:

You might want to remove a license from a controller, for example, if you have to take it out of service, for repairs, and want to use its license in a replacement unit. Follow the steps, below:

1. Install the dongle in your PC's USB port, and allow it to be recognized.
2. Start the Application Licensing Tool, using the following sequence:
Start → Programs → OpenBSI Tools → ControlWave Tools → Application Licensing

Application Parameters

The Application Parameters page is accessible from the **"Application Parameters"** tab in the Flash Configuration Utility. For instructions on calling up the Flash Configuration Utility, see *Flash Configuration – An Overview*, later in this manual. These parameters apply only to ControlWave-series units that are configured as IP nodes.

Note:

Unless you are using redundancy or have a specific need to edit these parameters (for example, if you are running an application with special memory requirements, or if you are encountering performance problems related to CPU activity), leave these parameters at their defaults. Users should exercise particular caution when modifying the application parameters for memory. Making a significant change to these parameters without understanding how the parameters interact could actually reduce the amount of available memory, even though you have increased the values of the parameters. When changing these parameters, use **only** small incremental changes.

The screenshot shows the 'Application Parameters' tab selected. The 'CPU' section contains three input fields: 'Goal Idle' set to 30%, 'Idle Min Ticks' set to 5, and 'Minimum Idle' set to 2%. The 'Memory' section contains three input fields: 'Prog.RAM' set to 850 KB, 'Data RAM' set to 10 KB, and 'Retain RAM' set to 30 KB. The 'Redundancy Transfer' section contains two IP address input fields, both set to 0.0.0.0.

CPU:

- Goal Idle** This is a goal expressing the percentage of time the ControlWave CPU should be idle. The default value is 30%. If this goal cannot be met, the DEFAULT task period will automatically be adjusted to free up CPU time.
- Idle Min Ticks** This is the minimum number of 1 millisecond clock ticks to be left between executions of the DEFAULT task. The default value is 2.
- Minimum Idle** If this percentage of free CPU time is not maintained within the ControlWave CPU, an overload exception is reported. The default value is 5%.

Memory:

Prog RAM	In kilobytes, this is the amount of memory reserved (at system start) for storing the generated code for the ControlWave project. This value should NOT be set significantly larger than the amount of memory actually needed for the project, because any unused reserved memory will be unavailable for data or other purposes. NOTE: If the system does not have sufficient memory to hold the user requests, the requests will be reduced proportionally. This defaults to 1024k in the ControlWave and ControlWave MICRO, and 256k in the ControlWaveLP. This can range from 10k to 1024k.
Data RAM	This is the size of storage reserved for variables in kilobytes. This can range from 10k to 1024k. This defaults to 256k in the ControlWave and ControlWave MICRO, and 64k in the ControlWaveLP. NOTE: This amount does not include historical data (audit/archive).
Retain RAM	This is the size of storage space (in kilobytes) reserved at system start for variables marked as 'RETAIN'. The values of variables marked as 'RETAIN' are preserved in the event of a warm start download. This can range from 0k to 1024k, and defaults to 256k in the ControlWave and ControlWave MICRO, and 64k in the ControlWaveLP.

Redundancy Transfer

Unit A Addr	This must be an IP address corresponding to an Ethernet port on the A controller in a redundant pair.
Unit B Addr	This must be an IP address corresponding to an Ethernet port on the B controller in a redundant pair.

Variations when using the ControlWave I/O Expansion Rack

For the ControlWave I/O Expansion Rack, the 'Memory' and 'CPU' sections of the Application Parameters page are omitted, and a 'Timeouts' section is added.

The **"Power Fail Timeout"** determines how outputs of the I/O rack should be set when power is restored following a power failure, or under certain circumstances, during a restart following a CPU watchdog.

"Host Comm Loss Timeout" specifies how the outputs of the I/O Rack should be affected in the event of a communication failure with the host ControlWave controller.

For a full description of these options, please see the *ControlWave I/O Expansion Rack Quick Setup Guide* (part number D301423X012).

Archive Configuration

What Are Archive Files?

Archive information is saved at the ControlWave-series controller in structures called **archive files**. The archive files are essentially tables of data stored in rows and columns. Each row has a timestamp, and sequence numbers, associated with it, and each column is associated with a process variable. A row of process variable values, along with its associated timestamp and sequence numbers is called a **record**.

A record constitutes a 'snapshot' of the values for those variables at that particular time. Depending upon how you configure the archive file, the values saved may be just instantaneous values, or they may be the result of calculations performed on data received during the collection interval (integration, averaging, etc.)

Data is stored in the archive file either at a specified interval (periodic storage) or based on settings in the ARCHIVE function block.

Some Background - Archive Calculations (Weight Factors, Intervals, and Samples)

Optionally, calculations can be performed on the data prior to its being saved in the archive file. These calculations could be averaging, integration, etc.

The calculations performed using the Archive system are particularly useful in fluid flow applications, for fluids in either the gas or liquid state. The calculations are intended to assist in meeting the requirements of the *American Petroleum Institute Manual of Measurement Standards, Chapter 21*, by supporting some of the averaging techniques described in that document. If desired, these same calculations may be utilized for other applications (besides the fluid flow applications for which they were originally intended).

In the following explanations some mathematical terminology is used that is directly connected to the execution rate of the ARCHIVE function block. The Archive will have been declared to have a periodic type and an Interval of 1 day or less. Within this Periodic Interval the ARCHIVE function block will be executed at an execution interval established by the task execution.

Some of the calculations make use of a "Weight factor". The weight factors are set as input parameters of the ARCHIVE function block, and can be any REAL number values. The weight factors are used to control whether a sample is used in the calculation performed. The weight factors are typically used to control the number of samples included in the averaging calculations, so that, for example, Pressure or Temperature is only averaged when the weight factors indicate that it is valid to use the samples.

The averaging methods also provide an automatic switch from one type of averaging to another anytime after the start of a periodic interval. If a periodic interval begins with Weight Factor2 in a non-zero state then throughout the interval Weight Factor2 controls the averaging – samples are only used if Weight Factor2 is not zero. If the Interval begins with Weight Factor2 already zero then each sample is weighted using Weight Factor1, but

if at any time during the interval Weight Factor2 becomes non-zero, all samples taken with Weight Factor1 are discarded and a new average is started under control of Weight Factor2. For the rest of the interval Weight Factor2 is in control and Weight Factor1 has no more effect.

For example, if Weight Factor1 represents the delta time between ARCHIVE function block executions, and Weight Factor2 represents the delta flow time between ARCHIVE function block executions, then for those archive intervals where no flow time occurred, the result is a straight time average. For those archive intervals where flow time has occurred, the result is a flow time average.

Defining a Straight-time Average Archive

To set up a simple straight-time averaged archive, there are two methods available:

Method 1: Set Weight Factor1 to the delta time between ARCHIVE function block executions each time the ARCHIVE function block is executed. (Note: The delta time between each execution of the ARCHIVE function block must be dynamically calculated by user-created logic in your program. Do NOT simply enter a constant based on the rate of execution of the task, because that will not account for any slippage in execution time.)

Method 2: Set Weight Factor1 to 1.0 each time the ARCHIVE function block is executed. This provides an average based on the *number of samples* taken. (This is equivalent to a time-based average.)

Archive configuration involves four basic steps:

1. Define the archive file(s) using the Flash Configuration Utility. This includes specifying the number of rows (records) and columns, the types of calculations performed, etc.
2. Identify in ControlWave Designer, every variable you want to have archived. This is accomplished by including these variables in a LIST function block.
3. Create another LIST to hold the current archive record. (THIS IS OPTIONAL). This allows the current archive record data to be used within your ControlWave project; if you don't want to do this, skip this step.
4. Configure an ARCHIVE function block in your ControlWave project. This function must be executed frequently enough to collect an adequate number of samples.

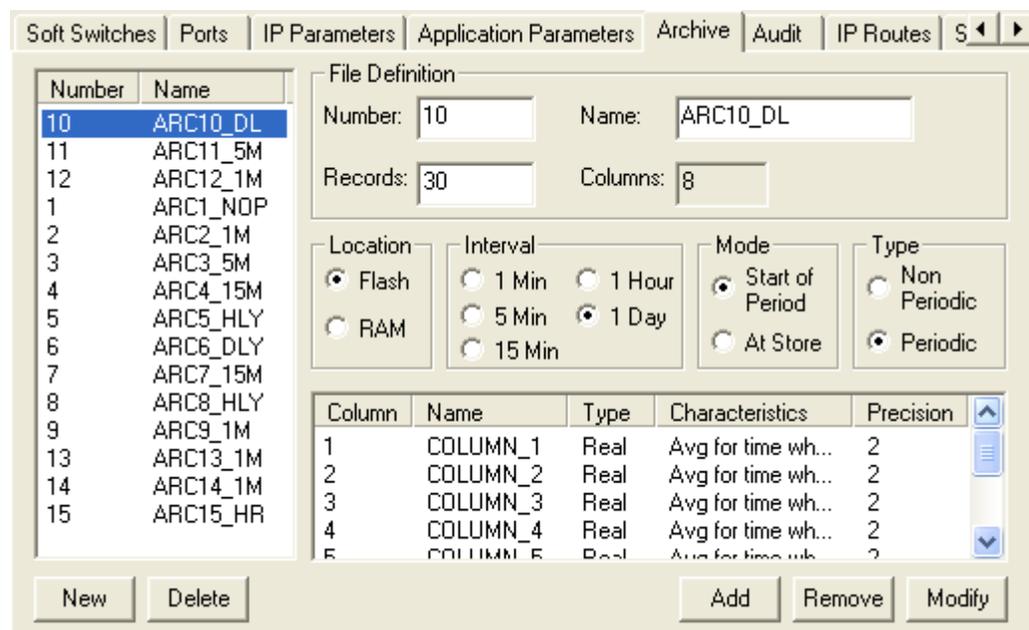
What can be done with the data from the Archive File(s)?

Once archiving has been fully configured, and the ControlWave project has been downloaded and has been running, archive data will be collected, and the Archive File(s) will be populated with data. There are various methods for extracting this Archive data:

- The Archive Collection web page control may be used to display archive data. This web page is included in the standard Web_BSI set, and allows you to display the archive data in Microsoft® Internet Explorer.
- The OpenBSI DataView utility can display archive data on the screen of your OpenBSI Workstation.
- The PocketBSI Data Viewer can display archive data on the PocketBSI AccessPack.
- The OpenBSI Harvester can collect the archive data, and store it in files on the OpenBSI Workstation, which the OpenBSI Data File Conversion Utility can export to human machine interface (HMI) packages such as OpenEnterprise.

Step 1. Define Archive Files(s) in the Flash Configuration Utility

The individual columns of the archive file, and various parameters that define how archiving is performed, are specified in the Archive page of the Flash Configuration Utility. Changes made in the Archive page will NOT take effect until the unit has been powered off and back on.



To begin defining an Archive File, click on the **[New]** button, then complete the fields as discussed, below:

File Definition

- Number** This is a unique ID number for this Archive File. It can range from 1 to 32767.
- Name** This is the archive file name. Up to 8 alphanumeric characters, beginning with a letter, can be used.
- Records** This determines how many rows of 'snapshot' data will be retained in this Archive File. For example, if you want to save 24 rows (records) enter 24 here. The upper limit on the number of records is based on the size of each record. The maximum size of an Archive File is fixed. This means that as the size of the archive record increases (based on number of columns, types of data, etc.) fewer records can be saved in the Archive File. NOTE: Each archive record includes 14 bytes to store the timestamp and sequence numbers, in addition to the bytes used to store the actual column data. Because the sizing of Archive Files can change based on firmware revision, consult the online help files for archives in ControlWave Designer. Also, there is a Histsize_Calculator.xls file included with OpenBSI if you need more detailed information on Archive File size.
- Columns** This is the number of columns in the Archive File. Each column corresponds to data for a particular variable. The number of columns can range from 1 to 64. Note: OpenBSI programs such as DataView and Harvester cannot collect archive records larger than 220 bytes. For all floating point (REAL) data, this means no more than 53 columns should be specified.

Location

- Flash** When selected, all Archive records will be stored in FLASH memory. FLASH memory is preserved in the event the ControlWave unit loses power, or if the unit's backup battery fails.
- RAM** When selected, all Archive records will be stored in static RAM. If the ControlWave unit is reset, for any reason, Archive records will be preserved only so long as the unit's backup battery continues to operate, or the user does not perform a system cold start. See the *Memory Usage* section for a discussion of system cold starts.

Interval

- 1 Min, 5 Min, 15 Min, 1 Hour, 1 Day** Only applies when the "Timestamp Mode" is **"Periodic"**. This specifies how often Archive records 'snapshots' should be stored.

Mode

- At Store** When **"At Store"** is chosen, the timestamp assigned to this archive record is

the time at which the record is stored.

Start of Period When "**Start of Period**" is chosen, the timestamp assigned to this archive record is the time at the beginning of the interval.

Type

Non-periodic When this is chosen, archive records are stored when the ARCHIVE function block executes, *if* the criteria determined by the iiMode terminal is met.

Periodic When this is chosen, archive records are stored when the ARCHIVE function block executes, *and* the chosen interval (either 1 minute, 5 minute, 15 minute, 1 hour, 1 day) has expired.

Column Definitions

To begin defining a column, click on the **[Add]** button. The Archive Column Definition dialog box will appear. Descriptions of the various fields are included, below; click on **[OK]** when finished defining the column, and you will return to the Archive page of the Flash Configuration Utility.

If you need to change the definition of a column after you've clicked on **[OK]**, click on the column number, then click on the **[Modify]** button, and the Archive Column Definition dialog box will be recalled.

If you need to delete a column you have defined, click on the number for the column, then click on the **[Remove]** button.

Title Is a description for the column. It can range from 1 to 16 characters.

Characteristics Determines the type of calculation to be performed on the collected data for this variable. Choose from the list box.

- Avg for time when WFactor2 != 0
- Arithmetic Mean Over WFactor1
- Avg of Sqrt(var) for time when WFacto
- Square of (average of sqrt(var))
- Instantaneous: Place value in Log**
- Min observed value for period
- Max observed value for period
- Place value in log, and 0 signal
- Integration over WFactor2

In these formulas, the following notation is used:

is the time at which the ARCHIVE function block executes and reads or 'samples' the variable.

is the number of module executions or samples that can occur within the defined Periodic Interval e.g., with a one second Task execution and a one Hour Periodic Interval "I" will be 3600.

The term 'Wfactor' used in these formulas refers to the Weight Factor. Weight Factors are specified in the ARCHIVE function block.

The choices are:

Avg for time when Wfactor2 !=0

This performs a simple sum and divide averaging calculation, but a weight factor is applied to each sample as it is read. The weight factor is set by other program logic, as required, to control the averaging done by the function block; it would typically be used to ensure that the variable being read is only averaged while another condition is valid. The equation is shown below:

$$\frac{\sum_{i=1}^n \text{Variable_Value}(i) * \text{WeightFactor2}(i)}{\sum_{i=1}^n \text{WeightFactor2}(i)}$$

(NOTE: This equation is only used when WeightFactor2 is non-zero.)

Arith Mean Over Wfactor1

Perform a simple sum and divide average with each sample weighted by WeightFactor1. See the equation below:

$$\frac{\sum_{i=1}^n \text{Variable_Value}(i) * \text{WeightFactor1}(i)}{\sum_{i=1}^n \text{WeightFactor1}(i)}$$

Avg of Sqrt(var) for time when Wfactor2 !=0

During the periodic interval, sample the variable, take the square root of the sample, multiply it by WeightFactor2, and sum it. At the end of the interval,

calculate the average square root and store the result in the Archive. See the equation, below:

$$\frac{\sum_{i=1}^n \sqrt{\text{Variable_Value}(i)} * \text{WeightFactor2}(i)}{\sum_{i=1}^n \text{WeightFactor2}(i)}$$

Sqr of (Avg of sqrt(var))

During the periodic interval, sample the variable, take the square root of the sample, multiply it by WeightFactor2 and sum it. At the end of the interval, calculate the average square root, then square it and store the result in the archive. The equation is shown below:

$$\left(\frac{\sum_{i=1}^n \sqrt{\text{Variable_Value}(i)} * \text{WeightFactor2}(i)}{\sum_{i=1}^n \text{WeightFactor2}(i)} \right)^2$$

Note: The result is zero if Weight Factor 2 is zero for the entire interval.

Instantaneous Place value in log

No calculation performed. At the end of the periodic interval, simply store the current value of the variable in the archive.

Min observed value for period

At the end of the periodic interval, store the lowest value of the variable among all values collected during this interval.

Max observed value from period

At the end of the periodic interval, store the highest value of the variable among all values collected during this interval.

Place value in log, and 0 signal

At the end of the periodic interval, store the current value of the variable, and

reset the variable to zero.

Integration over Wfactor2

Sum the samples taken during the periodic interval after multiplying each sample by WeightFactor 2. Perform the following calculation:

$$\sum_{i=1}^n \text{Variable_Value}(i) * \text{WeightFactor2}(i)$$

Data Type Choose any one of the data types. The default is Real. The other choices are: Boolean, Short Int, Int, Double Int, Long Int, Bit String-Byte, Bit String-Word, Bit-String-Dword.

Precision Specify the numerical precision in which values should be displayed.

Deleting An Existing Archive File Definition

To delete an existing archive file definition, which will also delete the columns and records of the archive file, click on the file name in the list box in the left part of the page, then click on the **[Delete]** button. NOTE: The actual file deletion does not take place until the ControlWave-series unit is powered off, and then re-started.

Working with String-Based Archives

See the ControlWave Designer online help for instructions on configuring string-based archives.

Step 2. In Your ControlWave Designer Project, Identify the variables you want to archive in the Archive List

All variables for which you would like to archive data must be included in the **Archive List**.

Important

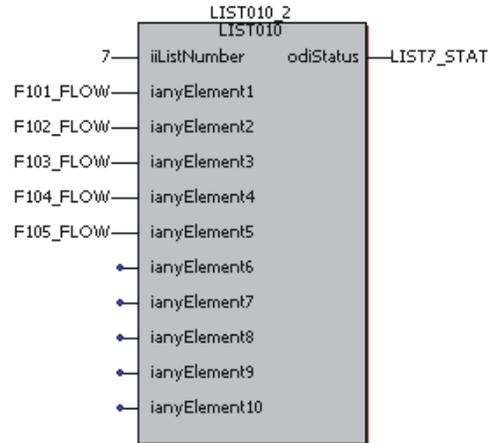
The variables in the Archive List must have a direct one-for-one correspondence with the columns of the archive file, as defined in the Archive page of the Flash Configuration Utility (Step 1). For example, the first variable in the archive list corresponds with column 1 of the archive file, the second variable in the archive list corresponds with column 2 of the archive file, etc.

Also note that the timestamp, global sequence number, and archive sequence number which are included at the beginning of every archive record, do not count as columns when specifying this correspondence.

To create the Archive List, insert one of the LIST function blocks (LIST010, LIST020, LIST030, LIST050, or LIST100) in your ControlWave Designer program. The choice of which LIST function block is determined by how many variables you want to include in the Archive List; for example, if you want to include 37 variables in your Archive List, you should choose LIST050, which can hold up to 50 variables; if you want to include 63 variables in your Archive List, you should choose LIST100, since it can hold up to 100 variables.

Note: Archive files cannot hold more than 64 columns of data, therefore, your archive list should not include more than 64 variables. The figure, at right, shows an Archive List with 5 variables.

You should insert the ARCHIVE function block in one of your POU's. The POU you choose must be part of a task which executes *faster* than the rate at which you want your archive calculations and storage to occur, since archive calculations and storage only occur when the ARCHIVE function block is executed.



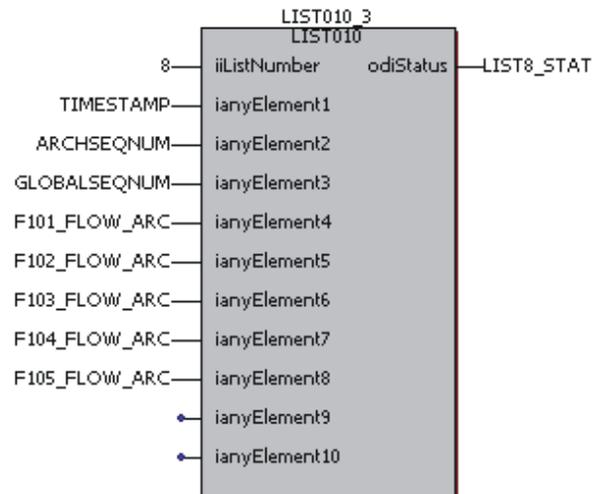
Notes:

- Typically, you would want to define control logic to only execute the list once. For information on how to do this, see the *Conditional Logic* section of this manual.
- All variables in your Archive List must be marked as 'PDD' in order to be collected by external archive collection programs such as DataView, or the Harvester.
- The variables you want to include in the Archive List must reside in the same POU as the LIST function block used for the Archive List, or they must be global variables.

Step 3. Create an Output List for Accessing the Most Recent Archive Record (OPTIONAL)

Optionally, you can create a list which will hold a specified Archive record. This allows the specified archive record data to be accessible within your ControlWave program.

The figure at right shows the Output list. The table, below, details the



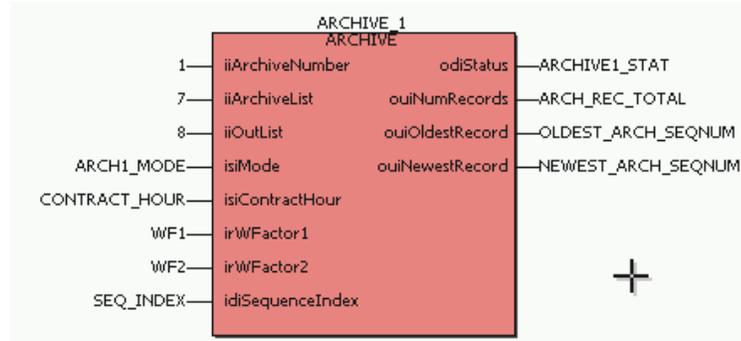
data type requirements of the output list used in this example.

Parameter Name	Suggested Variable Name	Variable Type	Value	Notes
iiListNumber	(no variable, entered as a constant)	INT	8	The list number (must be entered on the iiOutList parameter of the ARCHIVE function block)
odiStatus	LIST8_STAT	DINT	None	OPTIONAL – This reports status values regarding the Output List. Negative values indicate errors.
ianyElement1	TIMESTAMP	REAL	None	The timestamp associated with the archive record.
ianyElement2	ARCHSEQNUM	DINT	None	The local archive sequence number associated with the archive record.
ianyElement3	GLOBALSEQNUM	DINT	None	The global sequence number associated with the archive record.
ianyElement4 to ianyElement <i>n</i>	(desired variable names)	Match the data type to the corresponding variable in the Archive List	None	These elements of the list contain the values of the associated process variables for the archive record.

Step 4. Configure the ARCHIVE Function Block

Insert an ARCHIVE function block in one of your POU's of your project. The POU you choose must be part of a task which executes *fast enough to produce valid archive calculations*. If, for example, you want to calculate average flows for the past hour, for all variables in the Archive List, you must have executed the ARCHIVE function block enough times to obtain a valid average, based on your requirements. This might be once a minute, once every 10 seconds, etc. So, even though the calculation is *stored* hourly, the ARCHIVE function block must be executed faster.

When the ARCHIVE function block executes, it performs any intermediate calculations, and then only stores the archive data if the specified interval has been reached, or if on-demand archiving has been specified (MODE 2 or 3). Once the most recent archive data has been stored, it will be accessible in the Output List (if the Output List was configured).



The table, below, summarizes the configuration details for the parameters in the ARCHIVE function block for this particular example. More detailed information on the ARCHIVE function block is included in the on-line help.

Parameter Name	Variable Name	Variable Type	Value	Notes
iiArchiveNumber	(in this case we entered no variable, we used a constant instead)	INT	1	This must match the archive number defined for the archive file on the Archive page of the Flash Configuration Utility.
iiArchiveList	(no variable, entered as a constant)	INT	7	This must match the iiListNumber of the Archive List defined in Step 2.
iiOutList	(no variable, entered as a constant)	INT	8	This must match the iiListNumber of the Output List defined in Step 3.
isiMode	ARCH1_MODE	SINT	1	In this particular mode (Mode 1), archiving occurs at the interval specified for the archive file on the Archive page of the Flash Configuration Utility. This is the default mode; for information about other modes, see the on-line help.
isiContractHour	CONTRACT_HOUR	SINT	8	If interval configured on the web page is daily, this is the contract hour at which the daily collection should occur. This can range from 0 to 23.
irWFactor1	WF1	REAL	None	This weight factor is used in certain averaging calculations. This is discussed earlier in this section.

Parameter Name	Variable Name	Variable Type	Value	Notes
irWFactor2	WF2	REAL	None	This weight factor is used in certain averaging calculations. This is discussed earlier in this section..
idiSequenceIndex	SEQ_INDEX	DINT	None	This is only used in Mode 5 or 6. The user can specify the sequence number of a particular record, and its data will be reported in the output list (Mode 5), or the user can choose an indexed (fixed) position in the Archive file (Mode 6), and the data from that record will be reported in the output list. See the on-line help for the ARCHIVE function block for more info.
odiStatus	ARCHIVE1_STAT	DINT	None	OPTIONAL – This reports status values regarding the execution of the ARCHIVE function block. Negative values indicate errors.
ouiNumRecords	ARCH_REC_TOTAL	UINT	None	This is the total number of archive records currently in the archive file.
ouiOldestRecord	OLDEST_ARCH_SEQNUM	UINT	None	OPTIONAL – This reports the Local (Archive) Sequence number of the oldest record in the archive file.
ouiNewestRecord	NEWEST_ARCH_SEQNUM	UINT	None	OPTIONAL – This reports the Local (Audit) Sequence number of the newest record in the archive file.

Array Configuration

Instead of storing data exclusively in individual variables, you may find it more convenient, for some applications, to store some of the data in tabular form, such as in a **data array**. A data array is really variables which are part of a more complex structure. The structure is organized in rows and columns. It must be defined in the **"Data Types"** portion of the project tree, because it is considered to be a user-defined data type.

Arrays are most easily manipulated in POUs defined in the Structured Text (ST) language.

Defining an Array Data Type

Let's say, for example, that we wanted to save 3 temperature values (of type REAL), every hour, for an entire day. We need to define a data type for the columns of the array (which is type TEMPS), and an array of rows (which is type TODAYS_TEMPS). The 3 column by 24 row array is defined by entering the following in the **"Data Types"** section of the project tree:

```
TYPE
    TEMPS : ARRAY[1..3] OF REAL;
    TODAYS_TEMPS : ARRAY[1..24] OF TEMPS;
END_TYPE
```

Important

You should define this data type in your own data type worksheet. Do NOT use the SYS_VAR_WZ_TYPES sheet, because if you subsequently change your system variables, any data types you add to that sheet would be overwritten by the changes. To add your own data type worksheet, right-click on the 'Datatypes' item in the project tree, then choose **"Insert → Datatypes"** from the pop-up menus, then supply a name for the worksheet.

Creating an Array Using Your User-Defined Type

On one of your variable worksheets, create a variable using your newly defined data type. The text, below, creates an array called WEDNESDAY, which uses the data type we just defined.

```
VAR_EXTERNAL (* AUTOINSERT *)
    WEDNESDAY      :TODAYS_TEMPS;
END_VAR
```

Using the Array

In your structured text POU, you can assign values to elements of your array. Note that in the text shown, T1_TEMP, T2_TEMP, and T3_TEMP MUST be variables of type REAL, since each cell of the WEDNESDAY array holds a REAL value.

```
WEDNESDAY[1][1] := T1_TEMP;
```

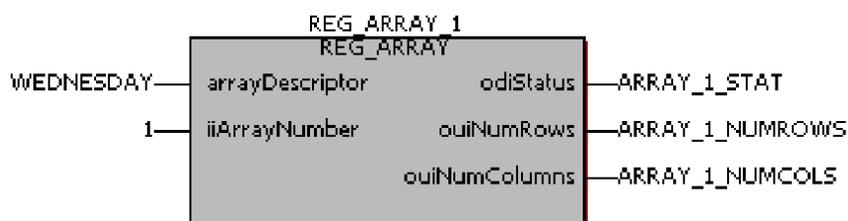
```
WEDNESDAY[1][2] := T2_TEMP;
```

```
WEDNESDAY[1][3] := T3_TEMP;
```

Making the Array Accessible to OpenBSI Collection Programs

In ControlWave Designer, data arrays are defined by a name. In the OpenBSI Utilities such as DataView, the Data Collector, and the Scheduler, however, data arrays are referred to by a number, since in Network 3000-series products, arrays only have numbers, not names.

To collect arrays from a ControlWave controller, using OpenBSI, the named array must be assigned a number using the REG_ARRAY function block. In addition, the array variable must be marked "PDD".



For this example, the following table details the usage of each parameter in the REG_ARRAY function block:

Parameter Name	Variable Name	Variable Type	Value	Notes
arrayDescriptor	WEDNESDAY	User-defined array type, in this case TODAYS_TEMPS	None	Indicates the name of your array, which is of some data type you defined.
iiArrayNumber	1 (just entered a constant here)	INT	1	Indicates the array number you are assigning to your array. When you collect this array using OpenBSI, this is the number which you use to refer to it in the OpenBSI utilities.
odiStatus	ARRAY_1_STAT	DINT	None	Reports status values regarding the REG_ARRAY function block. Negative values indicate errors.
ouiNumRows	ARRAY_1_NUMROWS	UINT	None	Indicates the number of rows in this array to be reported.
ouiNumColumns	ARRAY_1_NUMCOLS	UINT	None	Indicates the number of columns in this array to be reported.

Audit Configuration

Audit logging is one of the historical storage capabilities of the ControlWave-series controllers. It allows a record to be kept of significant events such as alarms, operator setpoint changes, operator logins, and system events.

There are three basic steps to configuration of Audit logging:

1. Define audit parameters on the 'Audit' page of the Flash Configuration Utility.
2. Define an Event list in your ControlWave project that lists non-alarm variables you want to monitor for changes.
3. Configure the AUDIT function block and execute it at the desired frequency.

What can be done with the data from the AUDIT data once it has been logged?

Once audit logging has been fully configured, and the ControlWave project has been downloaded and has been running, audit data will be collected, as events and alarms occur. There are various methods for extracting the Audit data:

- The Audit Collection web page control may be used to display audit data. This web page is included in the standard Web_BSI set, and allows you to display the audit data in Microsoft® Internet Explorer.
- The OpenBSI DataView utility can display audit data on the screen of your OpenBSI Workstation.
- The OpenBSI Harvester can collect the audit data, and store it in files on the OpenBSI Workstation, which the OpenBSI Data File Conversion Utility can export to human machine interface (HMI) packages such as OpenEnterprise.

Step 1. Set parameters in the Flash Configuration Utility

Configuration parameters for audit logging are set on the Audit page of the Flash Configuration Utility.

Note:

Changes made in the Audit page will NOT take effect until the unit has been powered off and back on.

Storage Location

- Flash** When selected, all Audit records will be stored in FLASH memory. FLASH memory is preserved in the event the ControlWave-series unit loses power, or if the unit's backup battery fails.
- RAM** When selected, all Audit records will be stored in static RAM. If the ControlWave-series unit is reset, for any reason, Audit records will be preserved only so long as the unit's backup battery continues to operate, or the user does not perform a system cold start. See the *'Memory Usage'* section for a discussion of system cold starts.

Logging Type

- Continuous** When Logging Type is specified as 'Continuous', if the storage area for audit records becomes full, the oldest records will be erased (overwritten) as new records come in.
- Stop on Full** When Logging Type is specified as 'Stop on Full', if the storage area for audit records becomes full, all logging will stop. NOTE: This does not have any impact on the variables themselves; they will continue to change, only their changes will not be logged to the Audit system.

Sizing

- Number of Events** Specifies the number of events to be logged. This value can range from 0 to 584. 0 is the default, which means that no events will be logged.
- Number of Alarms** Specifies the number of alarms to be logged. This value can range from 0 to 584. 0 is the default, which means that no alarms will be logged.

Number of Records in the Output Buffer [% of corresponding log]

When **"Flash"** is chosen as the storage location for the Audit Trail records, the Audit system cannot write data into the Alarm or Event logs when a flash log becomes full. Should alarm(s) or event(s) be generated during that time, it must be temporarily stored in the Output Buffer, until such time as the new useful records can be written into an Alarm or Event log in flash. The Output Buffer size specifies (as a percentage of the log file size) how much temporary storage is available during such operations. For example, if **"Number of Alarms"** is 300, and the **"Number of Records in the Output Buffer"** is set to 20, it means that up to 20% of 300 alarms (or 60 alarms) can be stored in the output buffer, before alarm data could be lost. If the **"Number of Alarms"** is 300 and **"Number of Records in the Output Buffer"** is set to 100, it means that 100% of the 300 alarms (or 300 alarms) can be stored in the output buffer, before alarm data could be lost. The default is 100.

Port**Logging Master Port**

Defines the only port which is capable of deleting the audit records from the audit logs. The Logging Master Port is only meaningful when the recording mode is set to **"Stop on Full"**. The port number for the logging master port can be one of the following values:

<u>Value</u>	<u>Port</u>
0	Serial Port 1
1	Serial Port 2
2	Serial Port 3
3	Serial Port 4
4	Serial Port 5
5	Serial Port 6
6	Serial Port 7
7	Serial Port 8
8	Serial Port 9
9	Serial Port 10
10	Serial Port 11
11-14	currently unused
	IP Port (any IP port on the unit)

Step 2. In ControlWave Designer, identify Variables for which you want to maintain Audit Logging

If there are several non-alarm variables for which you would like to maintain Audit logging, they must be included in the **Event List**. This is a list of variables which is scanned for any value changes, every time the AUDIT function block is executed.

Note

If you have a small number of event variables, instead of using the Event List, you can, if you choose, configure a separate AUDIT function block for each of the variables, and then assign each variable to the `ianyEventVar` parameter of its associated AUDIT function block.

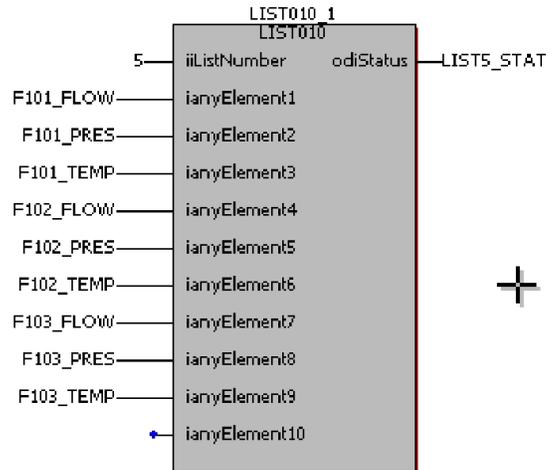
Important

- We strongly recommend you do NOT include in the Event List any variables tied to process I/O points or calculated variables which change frequently, because several minor fluctuations of a process I/O variable or calculated variable would generate multiple event records, thereby quickly filling up your event log. The Event List should be reserved for operator setpoints, configuration parameters, and other variables which change infrequently.
 - Alarm variables are automatically included in the Audit alarm log. The Audit alarm log, however, only includes the alarm messages generated when a variable enters its 'in-alarm' state, and when it returns to normal. Intermediate value changes to the alarm are NOT included in the alarm log. If you need to log this information, for example, for an operator setpoint variable, which is also configured as an alarm, you must include that variable in the event list. Again, however, this should only be done if the alarm changes infrequently. If you do not need this intermediate information, we recommend you do NOT include alarm variables in the Event List, since 'in alarm' and 'return to normal' messages are always stored in the Audit alarm log.
 - When the Audit alarm log and event log become full, they can be configured either to overwrite the oldest records when new data comes in, or to stop logging completely until one or more audit records are deleted by the user. You should configure the OpenBSI Harvester to periodically extract audit data and export it to your HMI software; to help prevent your audit logs from filling up.
 - DO NOT make on-line changes to the contents of the Event List as this will cause discrepancies in detection of value changes. If you want to change the Event List, make the changes off-line, then download the new project and execute a cold start of the unit.
-

To create the Event List, insert one of the LIST function blocks (LIST010, LIST020, LIST030, LIST050, or LIST100) in your ControlWave Designer program.

The choice of which LIST function block is determined by how many variables you want to include in the Event List; for example, if you want to include 15 variables in your Event List, you should choose LIST020, which can hold up to 20 variables; if you want to include 63 variables in your Event List, you should choose LIST100, since it can hold up to 100 variables.

The figure, at right, shows an Event List with 9 variables.



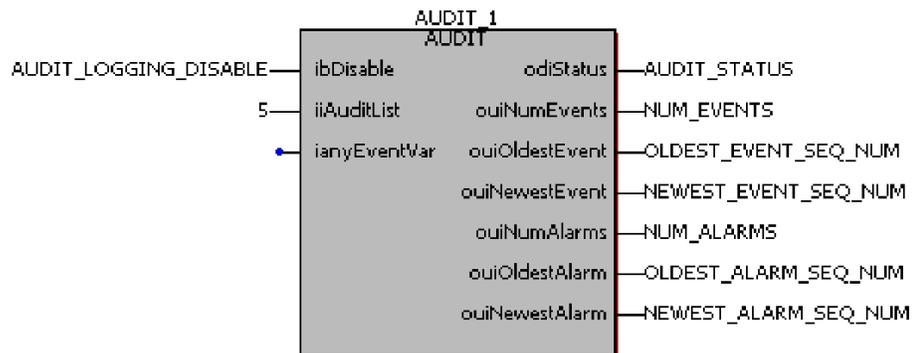
If larger lists are required, you can chain multiple LIST function blocks together by simply specifying the *same* **iListNumber** on each one.

Notes:

- All variables in your Event List must be marked as 'PDD' in order to be collected by external audit collection programs such as DataView, or the Harvester.
- The variables you want to include in the Event List must reside in the same POU as the LIST function block used for the Event List, or they must be global variables.
- Typically, you would want to define control logic to only execute the list once. See the 'Conditional Logic' section for more information on this subject.

Step 3. Configure an AUDIT Function Block

Insert an AUDIT function block into one of your POUs. The POU you choose must be part of a task which executes fast enough to handle your Audit logging requirements, since the Audit logging only occurs when the AUDIT function block is executed.



The following table summarizes the configuration details for the parameters in the AUDIT function block for this particular example. More detailed information on the AUDIT function block is included in the online help.

Parameter Name	Suggested Variable Name	Variable Type	Value	Notes
ibDisable	AUDIT_LOGGING_DISABLE	BOOL	FALSE	For disabling/ enabling audit logging.
iiAuditList	here, we have entered a constant of '5' since the LIST function block we configured in Step 2 is numbered '5'	INT	5	Identifies the iiListNumber of the LIST function block we are using.
ianyEventVar	left unused – only applies when there is no Event List. Used only when a single variable is to be monitored in the event log.	REAL, SINT, INT, or DINT	none	This parameter is only used if there is only one variable for which you want to maintain event logging. That single variable is identified by ianyEventVar.
odiStatus	AUDIT_STATUS	DINT	none	OPTIONAL – This reports status values regarding the execution of the AUDIT function block. Negative values indicate errors.
ouiNumEvents	NUM_EVENTS	UINT	none	OPTIONAL – This reports the number of events in the event log.
ouiOldestEvent	OLDEST_EVENT_SEQ_NUM	UINT	none	OPTIONAL - This reports the Local (Audit) Sequence number of the oldest event in the event log.
ouiNewestEvent	NEWEST_EVENT_SEQ_NUM	UINT	none	OPTIONAL - This reports the Local (Audit) Sequence number of the newest event in the event log.
ouiNumAlarms	NUM_ALARMS	UINT	none	OPTIONAL – This reports the number of alarms in the alarm log.
ouiOldestAlarm	OLDEST_ALARM_SEQ_NUM	UINT	none	OPTIONAL - This reports the Local (Audit) Sequence number of the oldest alarm in the alarm log.
ouiNewestAlarm	NEWEST_ALARM_SEQ_NUM	UINT	none	OPTIONAL - This reports the Local (Audit) Sequence number of the newest alarm in the alarm log.

BSAP Addressing and Networks

What is BSAP?

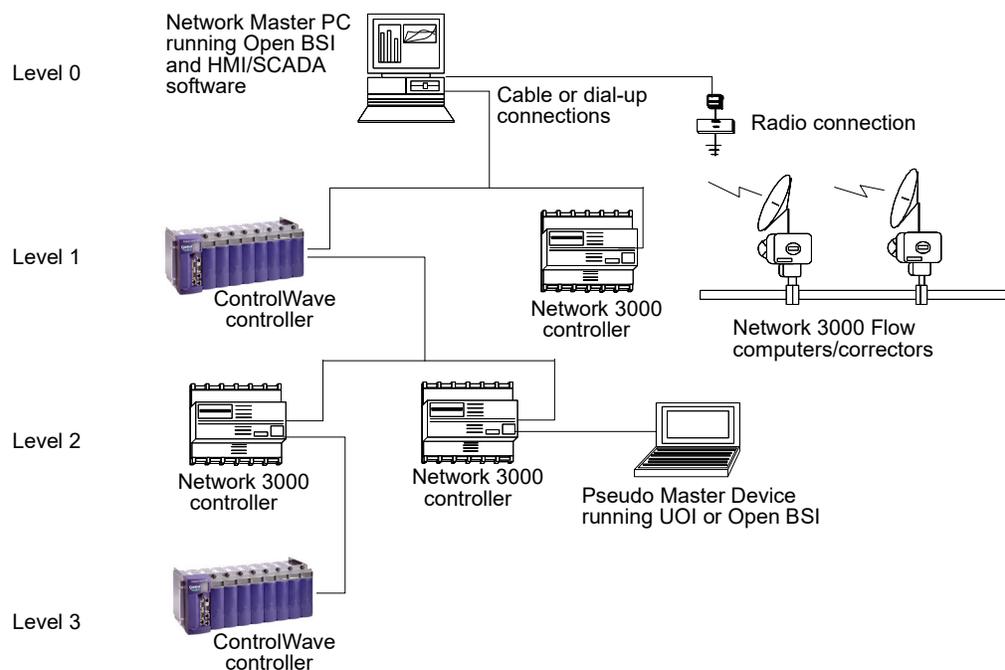
The Bristol Synchronous / Asynchronous Protocol (BSAP) is used for communication within ControlWave and Network 3000 controller networks. BSAP has been used in a wide variety of applications, and is particularly suited to networks where several controllers are connected via a multi-drop cable. It has also been used successfully with several different modes of data transmission including direct cable connections, dial-up modems, radios and satellite links.

At the top of a BSAP network is a host computer, called the **network master**. The network master is usually a PC workstation running **human-machine interface (HMI)** or **supervisory control and data acquisition (SCADA)** software such as Emerson OpenEnterprise™ software, or a third-party HMI package such as Intellution® FIX® or Iconics Genesis™. The HMI software communicates using the communications driver provided in the **Open Bristol System Interface (OpenBSI)** software. The HMI/SCADA software at the Network Master allows the operator to view what is going on in the network through graphical displays, trends, or printed logs and reports.

Note

Pseudo master devices can be connected to lower levels of the network to view data. These are similar to Network Masters, however, they are not considered to be “nodes” in the network, and so do NOT appear in the NETDEF files.

Below the Network Master are the remote process controllers.



The controllers in the network are organized into a hierarchical structure of one or more **levels**. A BSAP network can support up to six levels (not including the Network Master referred to as level 0.) The number of levels required varies depending upon the size and scope of your project.

Each controller (node) serves as a **master** to the nodes connected to it on the level immediately below, and as a **slave** to the node connected to it on the level immediately above. A node can have many slaves but only one master. Each master **polls** its slaves for data, which it retains in memory until it is polled by *its master*. In this way, data flows from slave to master, slave to master, etc. until it reaches the Network Master, where it is made accessible to the operator via HMI software.

Note

Initially, the ControlWave series could only serve as BSAP slave devices. Beginning with ControlWave firmware release CWP02.0, ControlWave-series controllers may also serve as BSAP master devices.

The user assigns each controller under a given master node a unique 7-bit **local address** (from 1 to 127). OpenBSI will automatically assign the controller a unique 15-bit global address (**GLAD**), based on its location in the network. Addresses and network structure are specified in the Network Definition (**NETDEF**) files generated by the OpenBSI **NetView** program. They must also be specified in the controller; either by switch settings (for certain Network 3000 controllers) or by parameters stored in FLASH memory (for ControlWave controllers, and certain Network 3000 controllers).

The network information stored internally by a node is called its **Node Routing Table (NRT)**. The NRT is updated whenever a valid time synchronization message (**TS/NRT**) message is received from the master node. Typically, this occurs when the Master is downloaded, but TS/NRT transmission can also be forced by the user via a menu selection in NetView.

The level of a given controller specifies how many intervening **communication lines** there are between it, and the network master. The first level controllers are called **top-level nodes** because data must travel over only 1 communication line to reach the Network Master. A communication line can consist of a direct cable connection, a radio or satellite link, or a dial-up modem connection. Each communication line is configured independently with its own **baud rate, poll period, timeout**, etc.

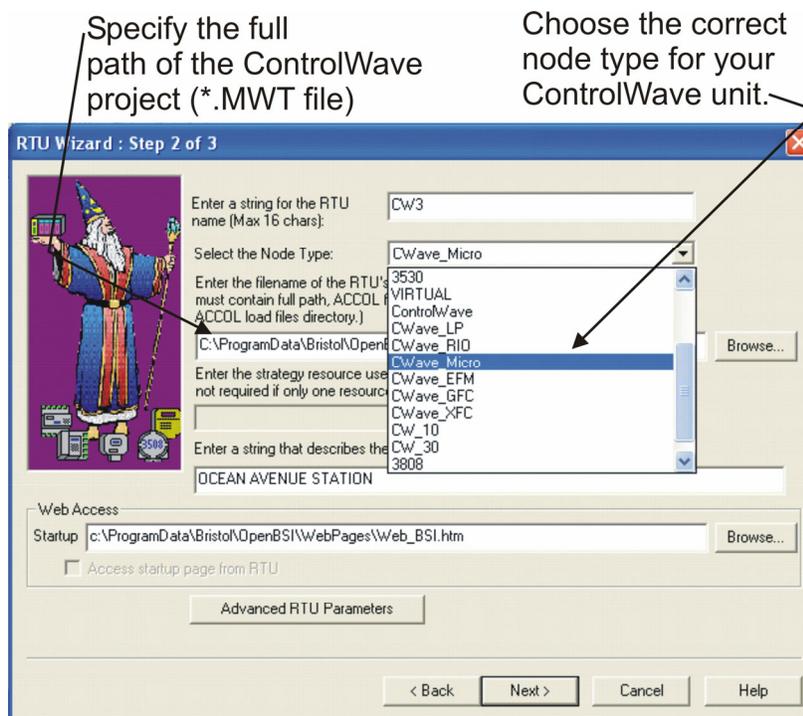
From a given node, BSAP **client/server communication** (transferring array or list data) is only possible to its Master node, any connected slave nodes, and any siblings (nodes on the same level which share the same master). If communication is required to any node not in these categories, it must be routed up using client / server function blocks (Master/Slave modules in Network 3000) at each individual level of the network, until it reaches either the Network Master, or a Master which is a sibling to another Master. The message can then be routed down, again, in the same way, until it reaches the desired node.

Within OpenBSI's NetView, the ControlWave can be added to an existing BSAP network in the same way as you would add any other controller.

Adding A ControlWave to an OpenBSI BSAP Network in the RTU Wizard

Simply choose the icon for the network to which you want to add the ControlWave, *right-click* on the icon, and choose **Add→RTU** to call up the RTU Wizard.

In the RTU Wizard, be sure you specify the appropriate node type (such as **ControlWave**, **CWave_LP** or **CWave_Micro**), and also specify the full path of the ControlWave project.



In addition, you can optionally specify the startup web page for the controller. Because this is a BSAP network, the startup web page must reside on the PC, and you must specify its full path. Web pages residing within the ControlWave are not accessible within a BSAP network so the **Access startup page from RTU** check box is NOT available.

You will also need to specify a local address for the ControlWave. The local address must match whatever local address you defined for the ControlWave on the 'Soft Switches' page of the Flash Configuration Utility. For information on configuring soft switches, see the discussion of the Flash Configuration Utility in Chapter 5 of the *OpenBSI Utilities Manual* (part D301414X012).

Full details on creating a BSAP network, and adding controllers to OpenBSI networks are included in Chapter 6 of the *OpenBSI Utilities Manual* (part D301414X012).

Tuning the BSAP Network

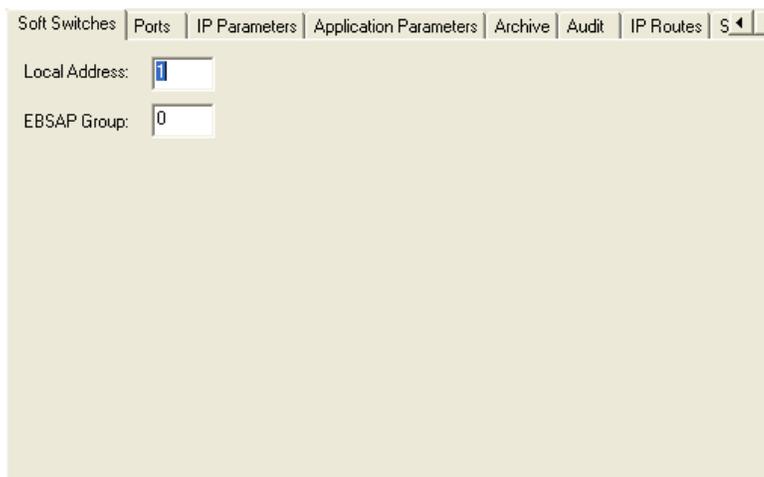
Instructions on tuning a BSAP network and troubleshooting communication problems are included in Chapter 14 of the *OpenBSI Utilities Manual* (part D301414X012). Additional information on BSAP networks is included in the *Network 3000 Communications Configuration Guide* (part D301413X012). For developers requiring information on the internal structure of BSAP networks, please see the *Network 3000 Communications Application Programmer's Reference*, (part D301401X012).

Setting the BSAP Local Address and EBSAP Group

The BSAP local address, and the expanded BSAP (EBSAP) group number are set on the 'Soft Switches' page of the Flash Configuration Utility.

BSAP Local Address

Every controller in a BSAP network has a “**Local address**” that ranges from 1 to 127, which is entered on the 'Soft Switches' page of the Flash Configuration Utility. This address identifies the controller's location within its level in the network, and is used for network routing. The local address entered here must match the local address specified for the RTU (controller) in OpenBSI's NetView program.



The screenshot shows the 'Soft Switches' page of the Flash Configuration Utility. The page has a tabbed interface with the following tabs: 'Soft Switches', 'Ports', 'IP Parameters', 'Application Parameters', 'Archive', 'Audit', and 'IP Routes'. The 'Soft Switches' tab is selected. Below the tabs, there are two input fields: 'Local Address:' with a value of '1' and 'EBSAP Group:' with a value of '0'. The background of the page is a light beige color.

EBSAP Group Number

If your network uses expanded BSAP, in which more than 127 nodes exist on the same BSAP network level, each controller is assigned to a particular expanded node addressing group. The group is identified by an “**EBSAP Group**” number, which is entered on the 'Soft Switches' page of the Flash Configuration Utility. For more information on EBSAP, please refer to the *Expanded BSAP (EBSAP) Communications* section of this manual. If you are NOT using Expanded node addressing (EBSAP) you MUST leave the “**EBSAP Group**” at 0.

For both the local address and EBSAP group number, changes will not take effect until after you have clicked [**Save to Rtu**], and powered the controller off and then and back on.

What is Client/Server Communication?

If desired, you can configure CLIENT and SERVER function blocks to transfer data arrays or lists from one controller to another.

A CLIENT function block requests array or list data from a SERVER function block in *another* controller. The SERVER function block processes the request, and sends the array or list data to the CLIENT.

Instructions for configuring the CLIENT / SERVER function blocks are included in the ControlWave Designer online help.

BSAP - Underlying Technical Details (For ADVANCED USERS)

This sub-section summarizes various aspects of BSAP. For a full explanation of BSAP messages, please see the *Network 3000 Communications Application Programmer's Reference* (part D301401X012).

Polling:

The polling function of the BSAP Master is cyclic. It is repeated at the rate specified by the `_P1_POLL_PER` system variable, in seconds. During a polling cycle all slave nodes belonging to the polling master are polled. If, for any reason, a complete pass cannot be completed within this period the next polling cycle is started immediately after the end of the current polling cycle. Only the nodes that are active, see `_SLAVE_POLL_DIS`, are considered for polling.

Poll and Response Sequence:

The BSAP Master sends the poll message to its slaves. When a slave node receives a poll message it takes one of the following actions:

1. It transmits an alarm message if one is waiting and the poll message has a flag that indicates that the Master will accept the alarm messages (polling for alarm), *or*
2. It transmits a data response message if one is waiting, *or*
3. It transmits an Acknowledgement with No Data to send protocol message.

Data Message Routing:

Local Messages: The local messages are the ones generated by an application, such as a BSAP Client/Server function block, in this node. Such messages arrive at the appropriate BSAP Master after the application sends these messages and the message routing has been completed. BSAP Master does not have to track such messages, as response messages are destined for local applications. Applications are responsible for performing the application response timeout.

Global Messages: The BSAP Slave may receive global messages that are addressed to the slaves in the network below this node. The message recipient performs the routing on these global messages and selects the appropriate Master for forwarding them to the

target node. The BSAP Master transmits these messages to the target nodes. A tracking structure is also created for each global message. If a response is not received to a tracked global message within the passthrough timeout (`_MSG_TIMEOUT`) the tracking structure is freed and any future response to that particular global message is discarded.

Response Messages:

The BSAP Master receives messages in response to the poll messages. It performs routing on these messages and they are forwarded to the proper destinations, local applications, BSAP Slave/Pseudo slaves, or Ethernet slaves.

Network Slave Port:

A ControlWave/ControlWaveLP port can have multiple slave ports which can be BSAP or IP. Among all of these ports, however, only one can serve as the **Network Slave Port**.

The Network Slave Port is the default route for upward traffic to global address (GLAD) 0 (Network Master).

To specify which port will serve as the Network Slave Port, the user must set the `_SLAVE_PORT` system variable to the port number.

TS/NRT Message:

Any Slave port can receive and process the Time Synchronization/Node Routing Table (TS/NRT) message. Separate system variables are available per serial port (`_Px_TS_DIS` and `_Px_NRT_DIS`) to allow each Slave to selectively determine whether it can or cannot process the Time Synch and/or NRT portion of the TS/NRT message. A flag is also available which causes the BSAP Slave to generate a request to its master for a TS/NRT message. All TS/NRT messages are accepted that are different than the current TS/NRT in this node. As the name implies the TS/NRT message is made up of two distinct entities:

1. **TIME SYNCH:** This part of the TS/NRT includes the complete system time and calendar information. When the Time Synch is processed the system time/calendar information is updated.
2. **NRT: Node Routing Table** - This part of the TS/NRT message is the heart of the BSAP message routing mechanism.

The BSAP Master sends a TS/NRT to its slave nodes as a result of the following:

- A new valid NRT has been received at any of the slave ports
- A slave node explicitly requests a TS/NRT message.
- After completion of a global download of one of its slaves. (Sends only to the node which received the download).

BSAP Master Port

Starting with release CWP02.0, the ControlWave-series of controllers support the BSAP Master mode of communication. The following functionality is possible:

- The ControlWave-series controller acts as the BSAP Master node to any ControlWave or Network 3000 controller with a BSAP Slave Port or Pseudo Slave port. This makes it possible for this node to be placed anywhere in the BSAP network, i.e. as a Network master or at any other level in the BSAP network.
- Multiple master ports are possible in the same controller.
- Each Master port can be assigned any consecutive numbered slaves from 1 to 127; e.g. slaves 1-10 to Master port 5, slaves 11-25 to Master port 2, and slaves 26-127 to Master port 4. **Note:** This slightly differs from Network 3000 configuration rules.
- It is possible to have gaps in the assigned slave numbers, e.g. slaves 1-5 to Master port 4, slaves 14-17 to Master port 5, and slaves 50-55 to Master port 2.
- A particular slave number cannot be assigned to more than one Master port, e.g. slaves 1-5 to Master port 2 and slaves 5-12 to Master port 4 would not work, because slave 5 is assigned to two different ports.
- Local download to the ControlWave-series controller is NOT supported.
- Global download of any Network 3000 slave controllers at any layer below a Master port is supported.
- BSAP Client/Server (similar to ACCOL II Peer-Peer) communication is supported.
- All BSAP global communication, including the *pass through* of report by exception (RBE) messages from the network below is supported.
- Supports all BSAP Alarm message communication from the network below. Alarm handling is as follows:
 - a. All global alarm report messages are logged with the local alarm system.
 - b. This ControlWave controller alarm system will forward these global alarm messages, interspersed with the local alarm reports, to all active alarm destinations.
 - c. If, at any time, the local alarm report pool becomes full the Master port will stop polling for alarms from the slave nodes.
 - d. Alarm polling will be resumed as soon as space is freed up in the local alarm report pool. Thus the loss of any alarm report from the slaves is prevented.
 - e. The global alarm reports carry the global network address of the reporting controller.
 - f. The ControlWave controller routes the global alarm acknowledgements, like any other global message, to the designated RTU.

Configuring A BSAP Master Port

BSAP Master Port configuration is divided into two parts:

- Configuring Flash Parameters
- Configuring System Variables for the Port

Note:

This section assumes your controller is either already part of a BSAP network (if you are using NetView), or that you established local communications using LocalView, and you have configured a local address, etc.

Configuring Flash Parameters

Step 1. In the Flash Configuration Utility, click on the 'Ports' tab, and choose the ControlWave port you want to configure (COM1, COM2, etc.) Then select 'BSAP Master' as the "Mode".

The screenshot shows the 'Ports' tab in the Flash Configuration Utility. On the left, a table lists available ports and their protocols:

Port	Protocol
COM1	BSAP Ma...
COM2	BSAP Sla...
COM3	UNUSED
COM4	BSAP Sla...
ENET1	IP
ENET2	UNUSED
ENET3	UNUSED

The right side of the window is divided into three sections:

- Physical Line Information:** Baud Rate: 115200, Bits Per Char: 8, Stop Bits: 1, Parity: NONE.
- Protocol:** Mode: BSAP Master, User Mode: 4096.
- Master Addressing:** Low Slave: 0, High Slave: 20.

Step 2. Enter the baud rate for the communication line in the "Baud Rate" field. 1200, 2400, 4800, 9600, 19200, 38400, 57600, or 115200 are all valid. The default is 9600.

Step 3. Define the range of BSAP local addresses used by the slave nodes of this BSAP master port. Enter the lower and upper ends of this range in the "Low Slave" and "High Slave" fields. These numbers must be integers in the range 1 to 127.

Step 4. Click on the [Save to Rtu] button, and respond to any sign-on prompts.

- Step 5. At this point, you can optionally make additional changes on other pages of the Flash Configuration Utility. When you are finished, turn off the ControlWave, then turn it back on, for the new port definition to come into effect.

Configuring System Variables

Within ControlWave Designer, start the System Variable Wizard by clicking on **View → System Variable Wizard**.

When the wizard has successfully established communications with ControlWave Designer, and your project is open, do the following:

1. Choose the 'Port Detail' tab.
2. Select the "Enable" box for the port which will serve as the BSAP Master.
3. Click **[Configuration]**.

First, check the box of the port you want to configure.

Next, click on the "Configuration" button for that port.

IP RBE Configuration	Data Line Monitor	Redundancy	Load Validation
System Configuration	System Information	System Statistics	Alarms
ID Strings	System Date/Time	Task Info	Port - Globals
			Port Detail
Port One <input checked="" type="checkbox"/> Enable Configuration Information RBE	Port Seven <input type="checkbox"/> Enable Configuration Information RBE		
Port Two <input type="checkbox"/> Enable Configuration Information RBE	Port Eight <input type="checkbox"/> Enable Configuration Information RBE		
Port Three <input type="checkbox"/> Enable Configuration Information RBE	Port Nine <input type="checkbox"/> Enable Configuration Information RBE		
Port Four <input type="checkbox"/> Enable Configuration Information RBE	Port Ten <input type="checkbox"/> Enable Configuration Information RBE		
Port Five <input type="checkbox"/> Enable Configuration Information RBE	Port Eleven <input type="checkbox"/> Enable Configuration Information RBE		
Port Six <input type="checkbox"/> Enable Configuration Information RBE			

4. In the Configuration page, select only the items shown in the following figure and enter appropriate values. A discussion of the various items appears below:

Once you select an item, you can specify its value in the corresponding field.

The items checked are used with BSAP Master ports.

Port 1 Configuration

Option	Parameter	Value
<input checked="" type="checkbox"/> Poll Time	_Px_POLL_PER	5
<input checked="" type="checkbox"/> Write Delay	_Px_WRITE_DEL	0
<input checked="" type="checkbox"/> Write (CTS) Timeout	_Px_WRITE_TMO	2500
<input type="checkbox"/> Ignore Echo Data	_Px_IGNORE_ECHO	FALSE
<input type="checkbox"/> Port Supports Dial	_Px_DIAL_PORT	FALSE
<input type="checkbox"/> Port performs Auto DTR shutdown	_Px_AUTO_DTR	FALSE
<input type="checkbox"/> BSAP Pad Front	_Px_PAD_FRONT	0
<input type="checkbox"/> BSAP Pad Back	_Px_PAD_BACK	0
Slave		
<input type="checkbox"/> Time Sync Disable	_Px_TS_DIS	FALSE
<input type="checkbox"/> Time Sync Needed	_Px_TS_FORCE	NO VAL
<input type="checkbox"/> Node Routing Table Disable	_Px_NRT_DIS	FALSE
<input type="checkbox"/> Alarm Disable	_Px_ALM_DIS	FALSE
<input type="checkbox"/> Immediate Response Disable	_Px_IMM_DIS	FALSE
<input type="checkbox"/> Fast Radio Interval	_Px_CYCLE_INT	0
<input type="checkbox"/> Fast Radio On Time	_Px_CYCLE_TIMEO	0
<input type="checkbox"/> Local Port	_Px_LOCAL_PORT	FALSE
<input type="checkbox"/> VSAT - Minimum Response Time	_Px_VSAT_MIN_RESP	0
<input type="checkbox"/> VSAT - Maximum Response Time	_Px_VSAT_MAX_RESP	0
Master		
<input checked="" type="checkbox"/> Retries	_Px_RETRIES	3
<input checked="" type="checkbox"/> Data Link Timeout	_Px_TIMEOUT	500
<input checked="" type="checkbox"/> Idle Polling	_Px_IDLE_POLL	FALSE
<input type="checkbox"/> VSAT - Up Ack Wait	_Px_VSAT_UP_ACK_WAIT	0
EBSAP Master		
<input type="checkbox"/> Max Slaves Under Virtual Nodes	_Px_MAX_SLAVES	
<input type="checkbox"/> Top Level Nodes	_Px_TOP_LEVEL_NODES	
<input type="checkbox"/> Total Slaves On Port	_Px_TOTAL_NODES	
<input type="checkbox"/> Array Number of Dead Array	_Px_DEAD_ARRAY	0
<input type="checkbox"/> Array Number of Disable Array	_Px_DISABLE_ARRAY	0

x = Port Number

OK Cancel

**Poll Time
(_Px_POLL_PER)**

This is the frequency (in seconds) at which the Master port will attempt to poll all of its slave nodes. For example, if the poll time is set to 30 seconds, then every 30 seconds, the Master port will attempt to poll all of its slaves nodes for data. If the Master port cannot complete a complete polling cycle within the specified poll time, it will start the new polling cycle as soon as it completes the current cycle.

Write Delay (<code>_Px_WRITE_DEL</code>)	This specifies a delay (in milliseconds) which must elapse before this master port attempts to communicate with its slave. This is useful if the slave hardware has a slower CPU-type (e.g. 186) and requires more turn-around time to respond to messages from its master.
Write (CTS) Timeout (<code>_Px_WRITE_TMO</code>)	Since the Clear-To-Send (CTS) must be received in order to transmit, this timeout (in milliseconds) is added to the expected message transmission time at the effective Baud Rate for this port. The message must be completely transmitted before the resulting timeout. The default is dynamic and calculated based on the current baud rate.
Retries (<code>_Px_RETRIES</code>)	This is the number of data link level retries if a transmission fails. The default is 0, i.e. only 1 transmission attempt, and no retries.
Data Link Timeout (<code>_Px_TIMEOUT</code>)	This is the data link level response timeout. Message transmission must <i>start</i> before expiration of this timeout.
Idle Polling (<code>_Px_IDLE_POLL</code>)	RESERVED FOR FUTURE USE.

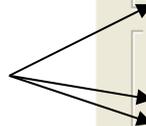
Click on **[OK]** when finished. Optionally, you can then click on the **[Information]** button in the Port Detail page, to specify variables used to store communication port statistics. (This page is not shown here).

Once you have configured all Port Detail parameters, you need to set global port parameters. Click on the 'Port - Globals' tab. Select only the items shown on the next page, and enter appropriate values. A discussion of the various items follows:

Passthru Timeout (<code>_MSG_TIMEOUT</code>)	All messages passing through the controller are tracked. This timeout applies to each passthrough message. If the value is ≤ 0 , the default timeout of 30000 milliseconds (30 seconds) is assumed.
Master - Dead Slaves (<code>_SLAVE_DEAD</code>)	This references an array of 127 BOOL variables. Array elements correspond to slave nodes of this Master Port. If a slave node is not responding to poll messages from the Master Port, its corresponding array element is set by the system to TRUE, and the node is declared 'dead'. This array is to report status.
Don't Poll Array (<code>_SLAVE_POLL_DIS</code>)	This references an array of 127 BOOL variables. Array elements correspond to slave nodes of this Master Port. Any element set to TRUE indicates that the corresponding slave node should NOT be polled by the Master Port. The default is FALSE. The user can turn off polling for a particular node by setting its corresponding array element to TRUE.

IP RBE Configuration	Data Line Monitor	Redundancy	Load Validation
System Configuration	System Information	System Statistics	Alarms
ID Strings	System Date/Time	Task Info	Port - Globals
		Port - Globals	Port Detail
Alarm			
<input type="checkbox"/> IP Alarm Retry Count	_ALM_RETRIES		3
<input type="checkbox"/> IP Alarm Retry Time - Active	_ALM_RET_ACT		60
<input type="checkbox"/> IP Alarm Retry Time - Dead	_ALM_RET_DEAD		120
Ethernet			
<input type="checkbox"/> Activity Timeout	_ETH_POLL_PER		0
<input type="checkbox"/> Port 1 Activity Flag	_ETH1_ACT		
<input type="checkbox"/> Port 2 Activity Flag	_ETH2_ACT		
<input type="checkbox"/> Port 3 Activity Flag	_ETH3_ACT		
IBP Processing			
<input type="checkbox"/> Ignore NRT from NHP	_NHP_IGNORE_NRT		FALSE
<input type="checkbox"/> Ignore Time Synch from NHP	_NHP_IGNORE_TS		FALSE
<input type="checkbox"/> Specify more than two NHPs	_NHP_ADDITIONAL_MASK		00000000
<input type="checkbox"/> Time Synch Delta Accuracy	_TS_DELTA_ACCURACY		0
Message Tracking			
<input checked="" type="checkbox"/> Passthru Timeout	_MSG_TIMEOUT		30000
BSAP			
<input type="checkbox"/> Slave Port number	_SLAVE_PORT		0
<input checked="" type="checkbox"/> Master - Dead Slaves	_SLAVE_DEAD		
<input checked="" type="checkbox"/> Don't Poll Array	_SLAVE_POLL_DIS		
<input type="checkbox"/> #NDARRY and #LINE sense flag	_BSAP_FLAG_SENSE		FALSE

Select these items when you configure a BSAP Master Port.



BSAP Slave Port

Configuring a BSAP Slave Port

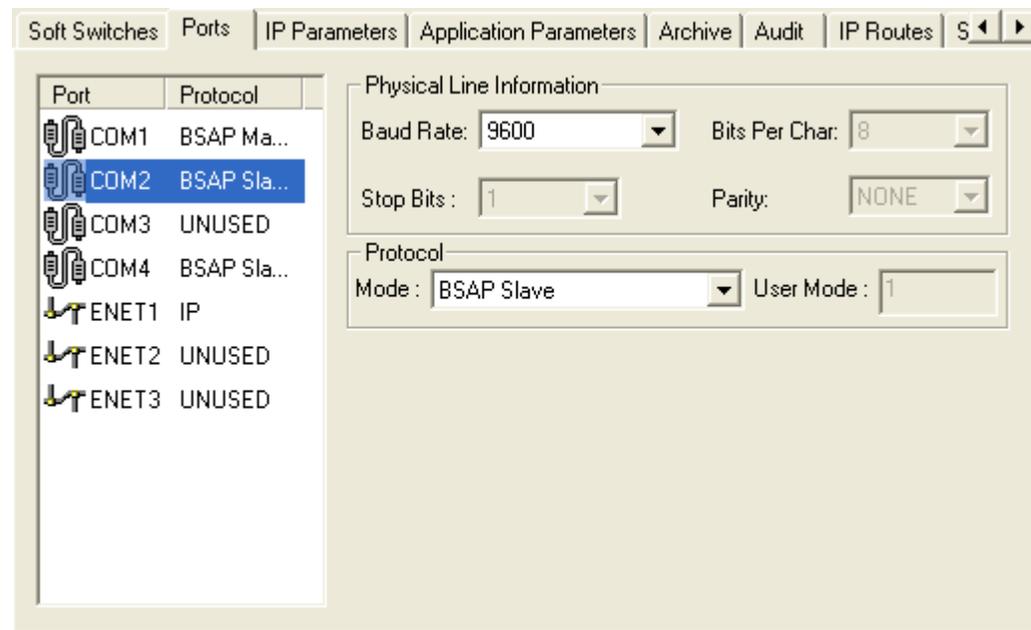
BSAP Slave Port configuration is divided up into two parts:

- Configuring Flash Parameters
- Configuring System Variables for the Port

Configuring Flash Parameters

Step 1. In the Flash Configuration Utility, click on the 'Ports' tab, and choose the ControlWave serial port you want to configure (COM1, COM2, etc.) Then select 'BSAP Slave' as the **"Mode"**.

Enter the baud rate for the communication line in the **"Baud Rate"** field. 1200, 2400, 4800, 9600, 19200, 38400, 57600, or 115200 are all valid. The default is 9600.



Step2. Click on the **[Save to Rtu]** button, and respond to the sign-on prompts.

Step 3. Turn off the ControlWave, then turn it back on for the new port definition to come into effect.

Configuring System Variables

Within ControlWave Designer, start the System Variable Wizard by clicking on **View → System Variable Wizard**.

When the wizard has successfully established communications with ControlWave Designer, and your project is open, do the following:

1. Choose the **Port Detail** tab.
2. Select the **"Enable"** box for the port which will serve as the BSAP Slave.
3. Click [**Configuration**].

First, check the box of the port you want to configure.

Next, click on the "Configuration" button for that port.

The screenshot shows the 'Port Detail' tab of the System Variable Wizard. It features a grid of 11 port configuration entries, labeled Port One through Port Eleven. Each entry includes an 'Enable' checkbox, a 'Configuration' button, and two other buttons labeled 'Information' and 'RBE'. Port Two is the only port where the 'Enable' checkbox is checked. Two arrows point from the text annotations to the 'Enable' checkbox and the 'Configuration' button of Port Two.

Port One	Port Seven
<input type="checkbox"/> Enable Configuration Information RBE	<input type="checkbox"/> Enable Configuration Information RBE
<input checked="" type="checkbox"/> Enable Configuration Information RBE	<input type="checkbox"/> Enable Configuration Information RBE
<input type="checkbox"/> Enable Configuration Information RBE	<input type="checkbox"/> Enable Configuration Information RBE
<input type="checkbox"/> Enable Configuration Information RBE	<input type="checkbox"/> Enable Configuration Information RBE
<input type="checkbox"/> Enable Configuration Information RBE	<input type="checkbox"/> Enable Configuration Information RBE
<input type="checkbox"/> Enable Configuration Information RBE	<input type="checkbox"/> Enable Configuration Information RBE
<input type="checkbox"/> Enable Configuration Information RBE	<input type="checkbox"/> Enable Configuration Information RBE
<input type="checkbox"/> Enable Configuration Information RBE	<input type="checkbox"/> Enable Configuration Information RBE

Once you select an item, you can specify its value in the corresponding field.

Check these items for a BSAP Slave Port.

Item	Parameter	Value
<input checked="" type="checkbox"/> Poll Time	_Px_POLL_PER	5
<input checked="" type="checkbox"/> Write Delay	_Px_WRITE_DEL	0
<input checked="" type="checkbox"/> Write (CTS) Timeout	_Px_WRITE_TMO	2500
<input type="checkbox"/> Ignore Echo Data	_Px_IGNORE_ECHO	FALSE
<input type="checkbox"/> Port Supports Dial	_Px_DIAL_PORT	FALSE
<input type="checkbox"/> Port performs Auto DTR shutdown	_Px_AUTO_DTR	FALSE
<input type="checkbox"/> BSAP Pad Front	_Px_PAD_FRONT	0
<input type="checkbox"/> BSAP Pad Back	_Px_PAD_BACK	0
Slave		
<input checked="" type="checkbox"/> Time Sync Disable	_Px_TS_DIS	FALSE
<input checked="" type="checkbox"/> Time Sync Needed	_Px_TS_FORCE	NO VAL
<input checked="" type="checkbox"/> Node Routing Table Disable	_Px_NRT_DIS	FALSE
<input checked="" type="checkbox"/> Alarm Disable	_Px_ALM_DIS	FALSE
<input type="checkbox"/> Immediate Response Disable	_Px_IMM_DIS	FALSE
<input type="checkbox"/> Fast Radio Interval	_Px_CYCLE_INT	0
<input type="checkbox"/> Fast Radio On Time	_Px_CYCLE_TIMED	0
<input type="checkbox"/> Local Port	_Px_LOCAL_PORT	FALSE
<input type="checkbox"/> VSAT - Minimum Response Time	_Px_VSAT_MIN_RESP	0
<input type="checkbox"/> VSAT - Maximum Response Time	_Px_VSAT_MAX_RESP	0
Master		
<input type="checkbox"/> Retries	_Px_RETRIES	3
<input type="checkbox"/> Data Link Timeout	_Px_TIMEOUT	500
<input type="checkbox"/> Idle Polling	_Px_IDLE_POLL	FALSE
<input type="checkbox"/> VSAT - Up Ack Wait	_Px_VSAT_UP_ACK_WAIT	0
EBSAP Master		
<input type="checkbox"/> Max Slaves Under Virtual Nodes	_Px_MAX_SLAVES	
<input type="checkbox"/> Top Level Nodes	_Px_TOP_LEVEL_NODES	
<input type="checkbox"/> Total Slaves On Port	_Px_TOTAL_NODES	
<input type="checkbox"/> Array Number of Dead Array	_Px_DEAD_ARRAY	0
<input type="checkbox"/> Array Number of Disable Array	_Px_DISABLE_ARRAY	0

x = Port Number

OK Cancel

4. In the Configuration page, select only the items shown in the figure above and enter appropriate values. A discussion of the various items appears, below:

Poll Time
(_Px_POLL_PER)

This defines a period of time (in seconds) during which this Slave node expects to receive a poll message from its Master. If a poll message

does not arrive within this period of time, the Master is assumed to have failed, and all messages queued to go up to the Master are discarded. If you are experiencing discards for transmission on this slave port (as reported by a nonzero value on the `_Px_DISC_TRANS` statistic system variable), it is recommended that you increase this value until no further discards occur. The default value of 5 should generally be increased to a larger number, e.g. 20, especially, if you are experiencing this problem.

Write Delay
(`_Px_WRITE_DEL`)

This specifies a delay (in milliseconds) which must elapse, before this Slave port attempts to communicate with its Master. This is useful if the Master hardware has a slower CPU-type (e.g. 186) and requires more turn-around time to accept responses from the Slave.

Write (CTS) Timeout
(`_Px_WRITE_TMO`)

Since the Clear-To-Send (CTS) must be received in order to transmit, this timeout (in milliseconds) is added to the expected message transmission time at the effective Baud Rate for this port. The message must be completely transmitted before the resulting timeout. The default is dynamic and calculated based on the current baud rate.

Time Sync Disable
(`_Px_TS_DIS`)

When set to TRUE, any time synchronization (TimeSync) message arriving at this slave port will be ignored. The default is FALSE which means TimeSync messages received at this port will be accepted and processed.

Time Sync Needed
(`_Px_TS_FORCE`)

When set to TRUE, the Slave sends a request to the Master Port for a TimeSync/Node Routing Table (TS/NRT) message. Once the TS/NRT is received, this is cleared.

Node Routing Table Disable
(`_Px_NRT_DIS`)

When set to TRUE, any Node Routing Table (NRT) message arriving at this slave port will be ignored. The default is FALSE which means NRT messages received at this port will be accepted and processed.

Alarm Disable
(`_Px_ALM_DIS`)

This allows you to disable alarm transmissions through this port.

Click **[OK]** when finished.

Identifying the Network Slave Port:

Although you may define multiple Slave Ports in your controller, only one of these ports can be the **Network Slave Port**. The Network Slave Port is the only port, among all the serial Slave Ports and IP ports, that is defined as the upward route for message traffic to the Network Master. To identify this Slave Port as the Network Slave Port, click on the 'Port - Globals' tab. Select only the item shown in the figure, below, and enter the port number of the Network Slave Port.

Valid entries for the Network Slave Port are:

- 1 = COM1
- 2 = COM2
- 3 = COM3
- 4 = COM4
- 5 = COM5
- 6 = COM6
- 7 = COM7
- 8 = COM8
- 9 = COM9
- 10 = COM10
- 11 = COM11
- 12 = currently unused
- 13 = IP (covers Ethernet as well as PPP serial ports)
- 14 = IP (covers Ethernet as well as PPP serial ports)
- 15 = IP (covers Ethernet as well as PPP serial ports)

IP RBE Configuration	Data Line Monitor	Redundancy	Load Validation	
System Configuration	System Information	System Statistics	Alarms	System Flags
ID Strings	System Date/Time	Task Info	Port - Globals	Port Detail

Alarm			
<input type="checkbox"/> IP Alarm Retry Count	_ALM_RETRIES		3
<input type="checkbox"/> IP Alarm Retry Time - Active	_ALM_RET_ACT		60
<input type="checkbox"/> IP Alarm Retry Time - Dead	_ALM_RET_DEAD		120

Ethernet			
<input type="checkbox"/> Activity Timeout	_ETH_POLL_PER		0
<input type="checkbox"/> Port 1 Activity Flag	_ETH1_ACT		
<input type="checkbox"/> Port 2 Activity Flag	_ETH2_ACT		
<input type="checkbox"/> Port 3 Activity Flag	_ETH3_ACT		

IBP Processing			
<input type="checkbox"/> Ignore NRT from NHP	_NHP_IGNORE_NRT		FALSE
<input type="checkbox"/> Ignore Time Synch from NHP	_NHP_IGNORE_TS		FALSE
<input type="checkbox"/> Specify more than two NHPS	_NHP_ADDITIONAL_MASK		00000000
<input type="checkbox"/> Time Synch Delta Accuracy	_TS_DELTA_ACCURACY		0

Message Tracking			
<input type="checkbox"/> Passthru Timeout	_MSG_TIMEOUT		30000

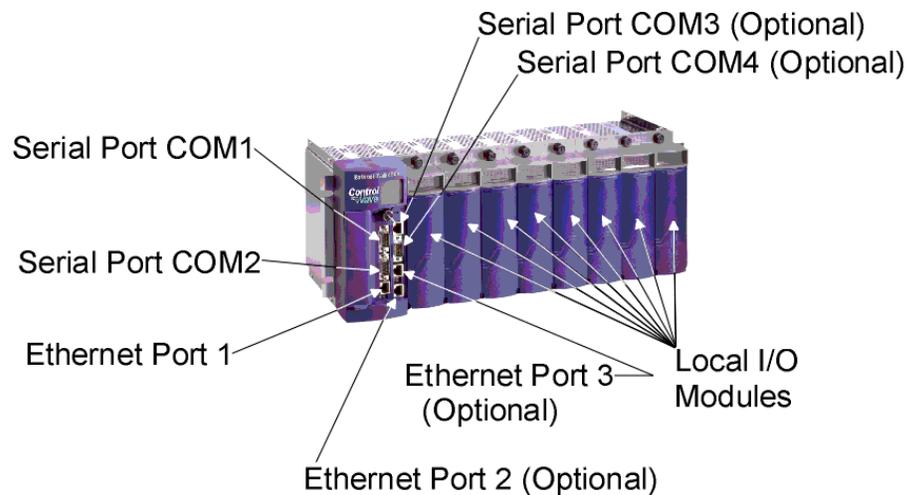
BSAP			
<input checked="" type="checkbox"/> Slave Port number	_SLAVE_PORT		2
<input type="checkbox"/> Master - Dead Slaves	_SLAVE_DEAD		
<input type="checkbox"/> Don't Poll Array	_SLAVE_POLL_DIS		
<input type="checkbox"/> #NDARRY and #LINE sense flag	_BSAP_FLAG_SENSE		FALSE

There can only be one Network Slave Port. Here we are choosing COM2 as the Network Slave Port.

Communication Ports

ControlWave Process Automation Controller, ControlWave Redundant Controller

Both the ControlWave Process Automation Controller, and the ControlWave Redundant Controller have from 2 to 4 serial communication ports, and from 1 to 3 Ethernet communication ports. (NOTE: The Redundant Version actually has twice the number of physical ports, but the second set serve in standby mode, unless there is a failure of the primary unit.) These ControlWave units have COM1 and COM2 (both serial). Depending upon the options you purchase, you may have up to two additional serial ports (COM3 and COM4) and from one to three Ethernet Ports (Ethernet Port 1, Ethernet Port 2 and Ethernet Port 3).



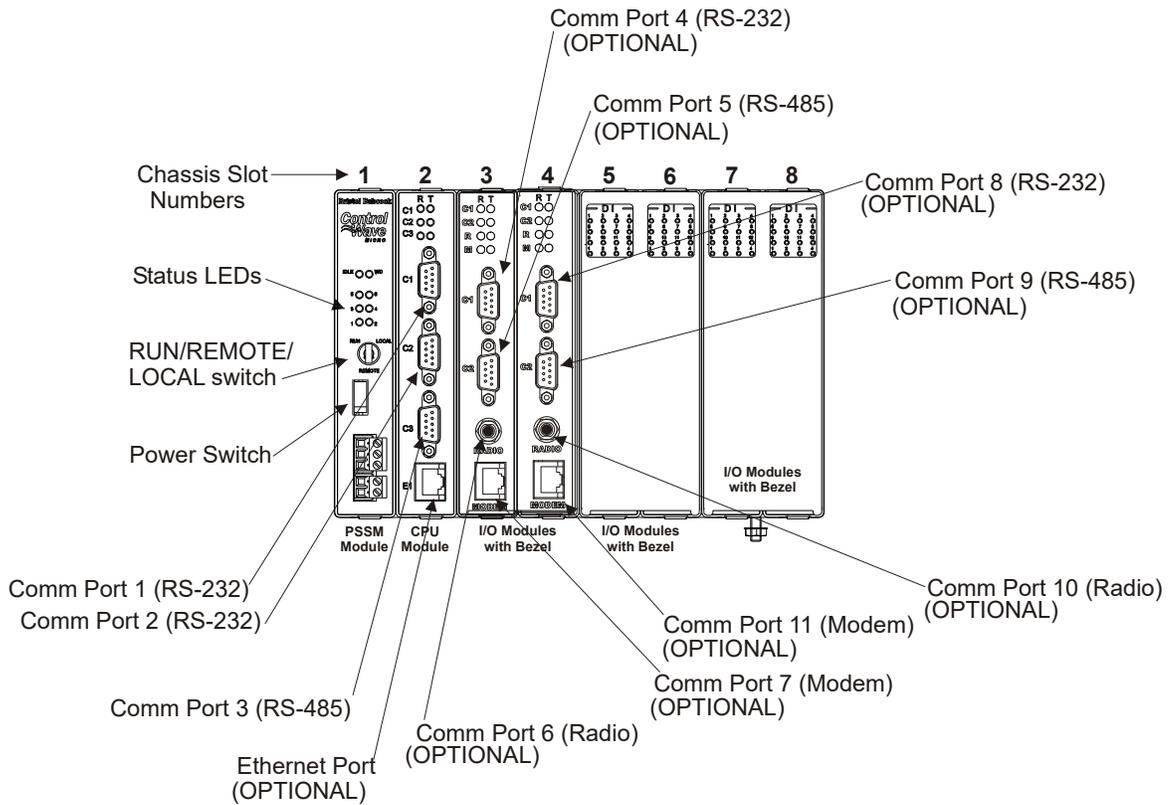
ControlWave Process Automation Controller

(ControlWave Redundant Controller Port locations are similar)

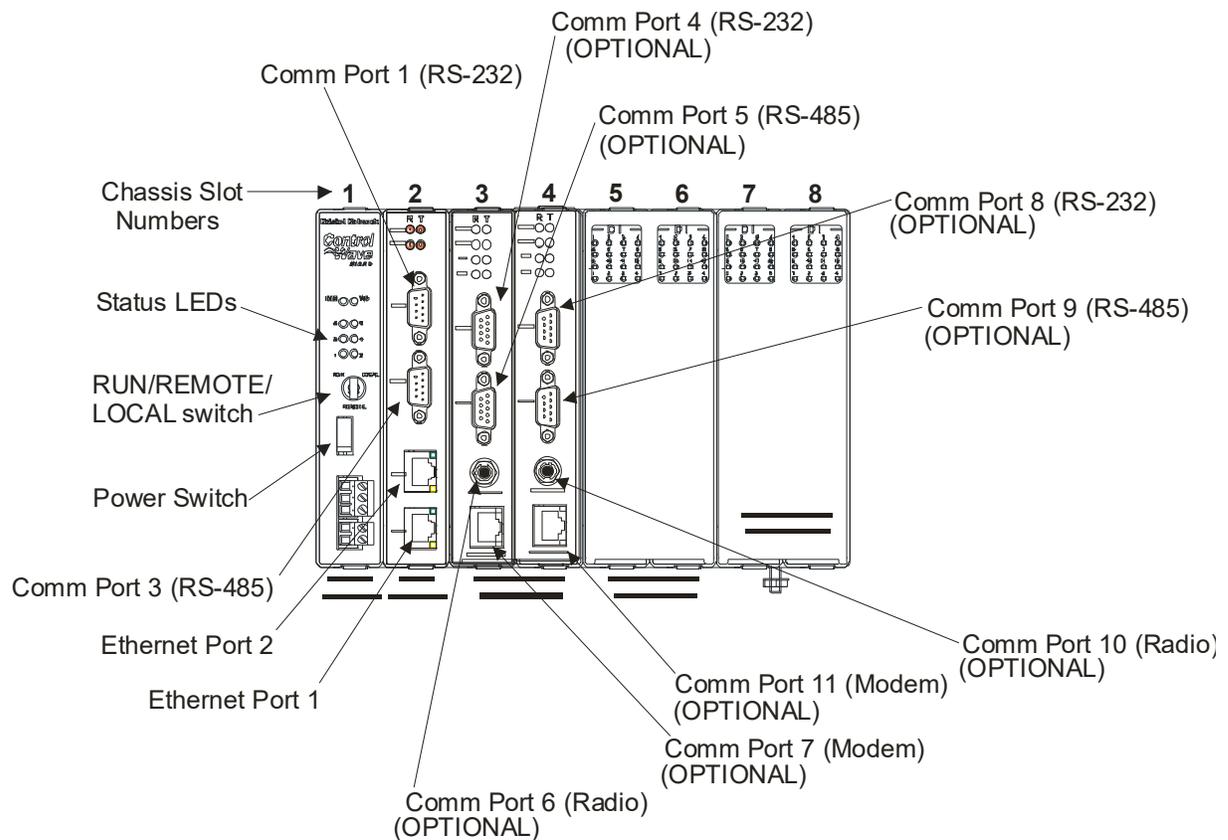
ControlWave MICRO Process Automation Controller

- The ControlWave MICRO Process Automation Controller CPU Module (CPU board) in Chassis Slot 2 may be ordered with different combinations of communication ports. One option allows for three serial ports on the CPU board where COM1 and COM2 are RS-232 and COM3 is RS485. Another option allows for the same three serial ports, with the addition of a single Ethernet port. A third available option for the CPU board is two serial ports, where COM1 is RS-232 and COM3 is RS-485 and two Ethernet ports; with this option, there is no COM2.
- If, instead of installing an I/O Module (board) in Chassis Slot 3, you install an Expansion Communication Module (ECOM board) in Chassis Slot 3, 4 additional serial ports are available. COM4 is RS-232, COM5 is RS-485, COM6 is for radio communication, and COM7 is for a modem. (OPTIONAL)

- If, instead of installing an I/O Module (board) in Chassis Slot 4, you install an Expansion Communication Module (ECOM board) in Chassis slot 4, another 4 additional serial ports are available. COM8 is RS-232, COM9 is RS-485, COM10 is for radio communication, and COM11 is for a modem. (OPTIONAL). *NOTE: This board can only be installed if there is already an Expansion Communication Module (ECOM board) in Chassis Slot 3; it cannot be installed if there is an I/O module in Chassis Slot 3.*



ControlWave MICRO controller with 1 Ethernet port



ControlWave MICRO controller with 2 Ethernet ports

ControlWave MICRO I/O Expansion Rack

The number and location of communication ports on the ControlWave MICRO I/O Expansion Rack matches those of the ControlWave MICRO.

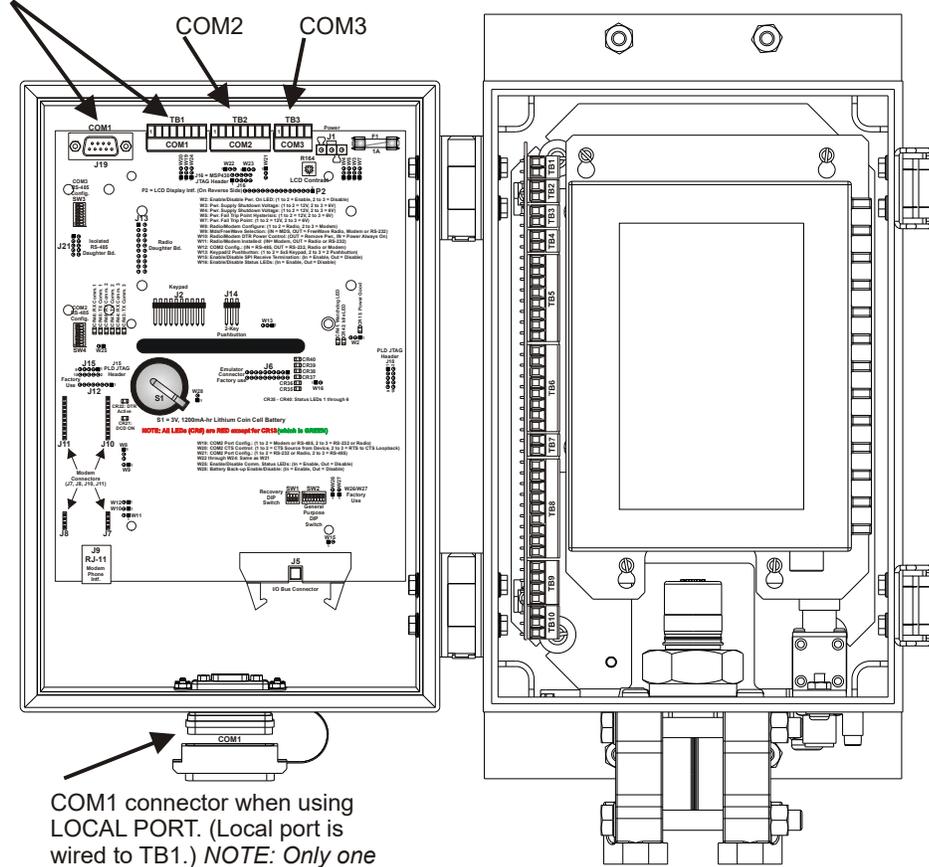
ControlWave Electronic Flow Meter (EFM)

- Communication port options for the EFM are identical to those of the ControlWave MICRO described on the previous page. The main difference is that EFM units may have fewer chassis slots.

ControlWave Gas Flow Computer Classic (GFC-CL)

- The ControlWave Gas Flow Computer Classic comes standard with three serial ports on the CPU board. COM1 is RS-232. COM2 can be RS-232 or RS-485, and COM3 is RS485. NOTE: COM1 has different connector types for you to choose from, depending on usage.
- The GFC-CL does NOT have any Ethernet ports, but serial IP can be done via PPP.

COM1 has a choice of different connectors (either of these, or the Local Port, below).

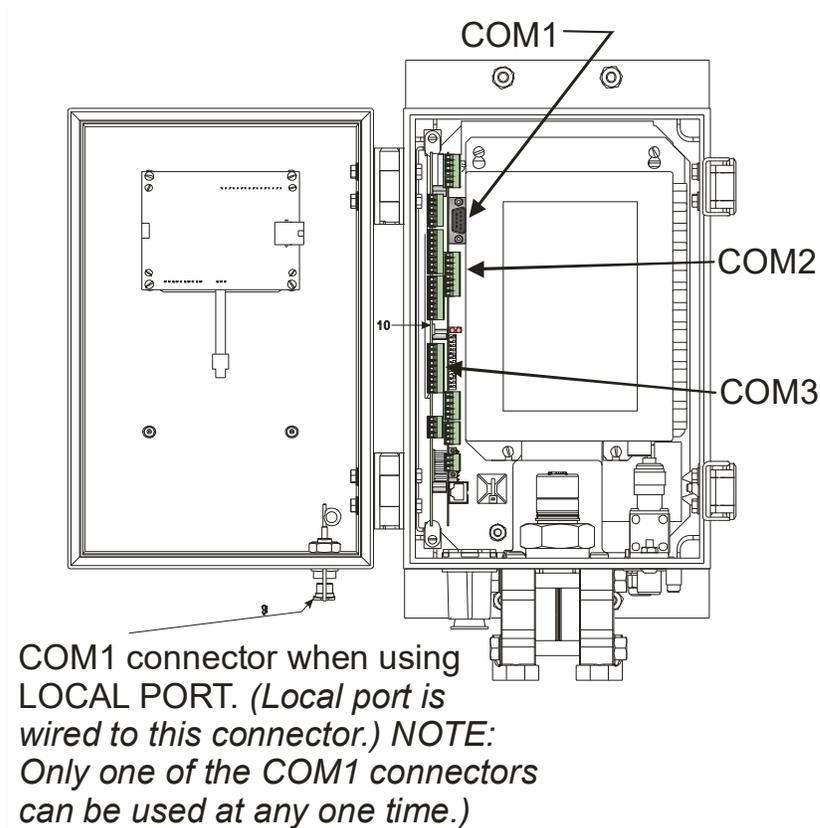


COM1 connector when using LOCAL PORT. (Local port is wired to TB1.) **NOTE: Only one of the COM1 connectors can be used at any one time.**

ControlWave GFC-CL

ControlWave Gas Flow Computer (GFC)

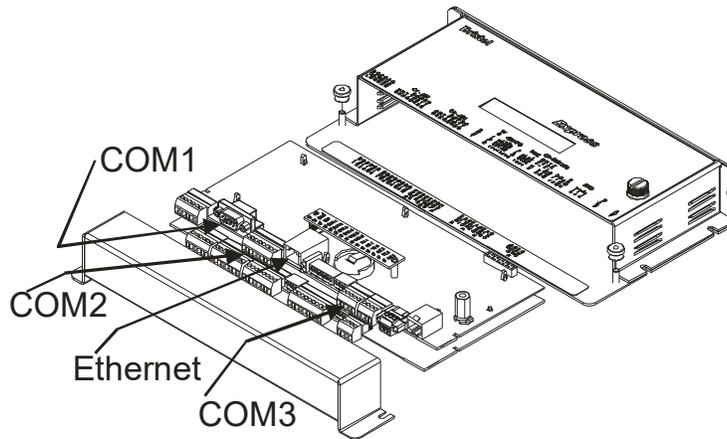
- The ControlWave Gas Flow Computer comes standard with three serial ports on the CPU board. COM1 and COM2 support RS-232 and COM3 can support either RS-232 or RS485. **Note:** COM1 has different connector types for you to choose from, depending on usage.
- Some models of the ControlWave GFC support an Ethernet port; for those that do not, serial IP can be done via PPP using the serial port(s).



ControlWave GFC

ControlWave Express

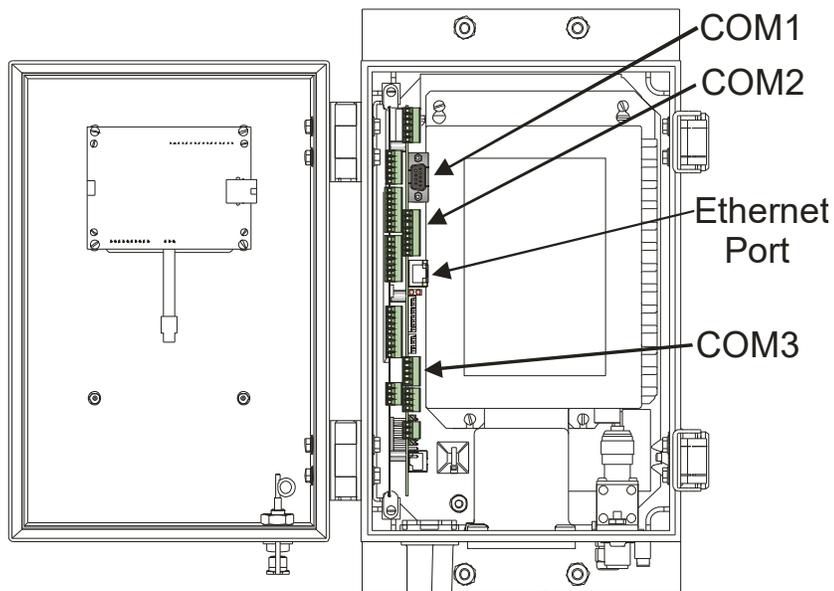
- The ControlWave Express comes standard with three serial ports on the CPU board. COM1 and COM2 support RS-232 and COM3 can support either RS-232 or RS485.
- Some models of the ControlWave Express support an Ethernet port; for those that do not, serial IP can be done via PPP using the serial port(s).



ControlWave Express

ControlWave Express PAC

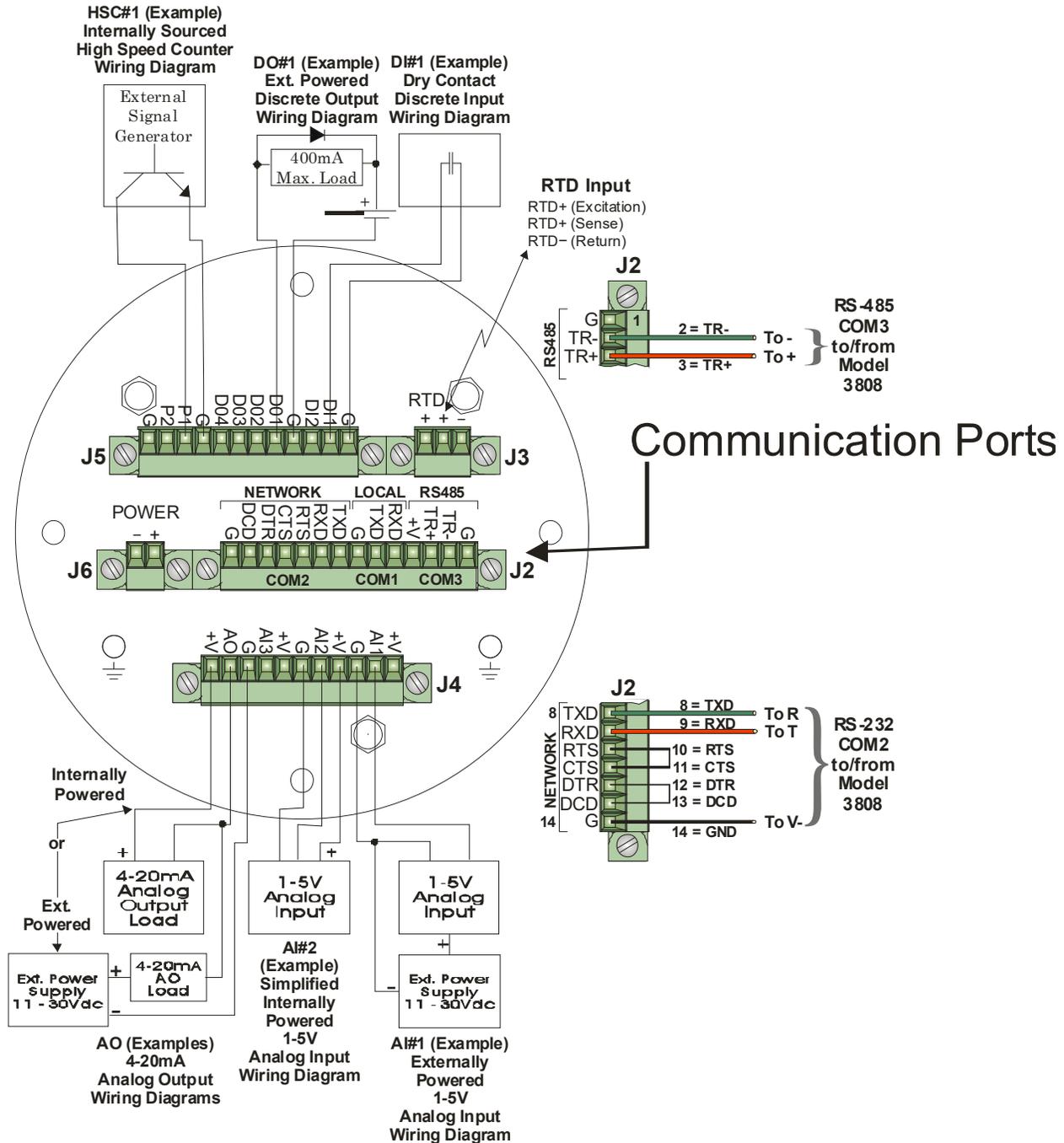
- The ControlWave Express PAC comes standard with three serial ports on the CPU board. COM1 and COM2 support RS-232 and COM3 can support either RS-232 or RS485.
- Some models of the ControlWave Express PAC support an Ethernet port; for those that do not, serial IP can be done via PPP using the serial port(s).



ControlWave Express PAC

ControlWave Explosion-Proof Flow Computer (XFC)

The ControlWave Explosion-Proof Flow Computer's communication ports are wired inside the cover with wires brought in through a conduit; there are no external connectors due to the explosion-proof rating.



ControlWave XFC

ControlWave CW_10

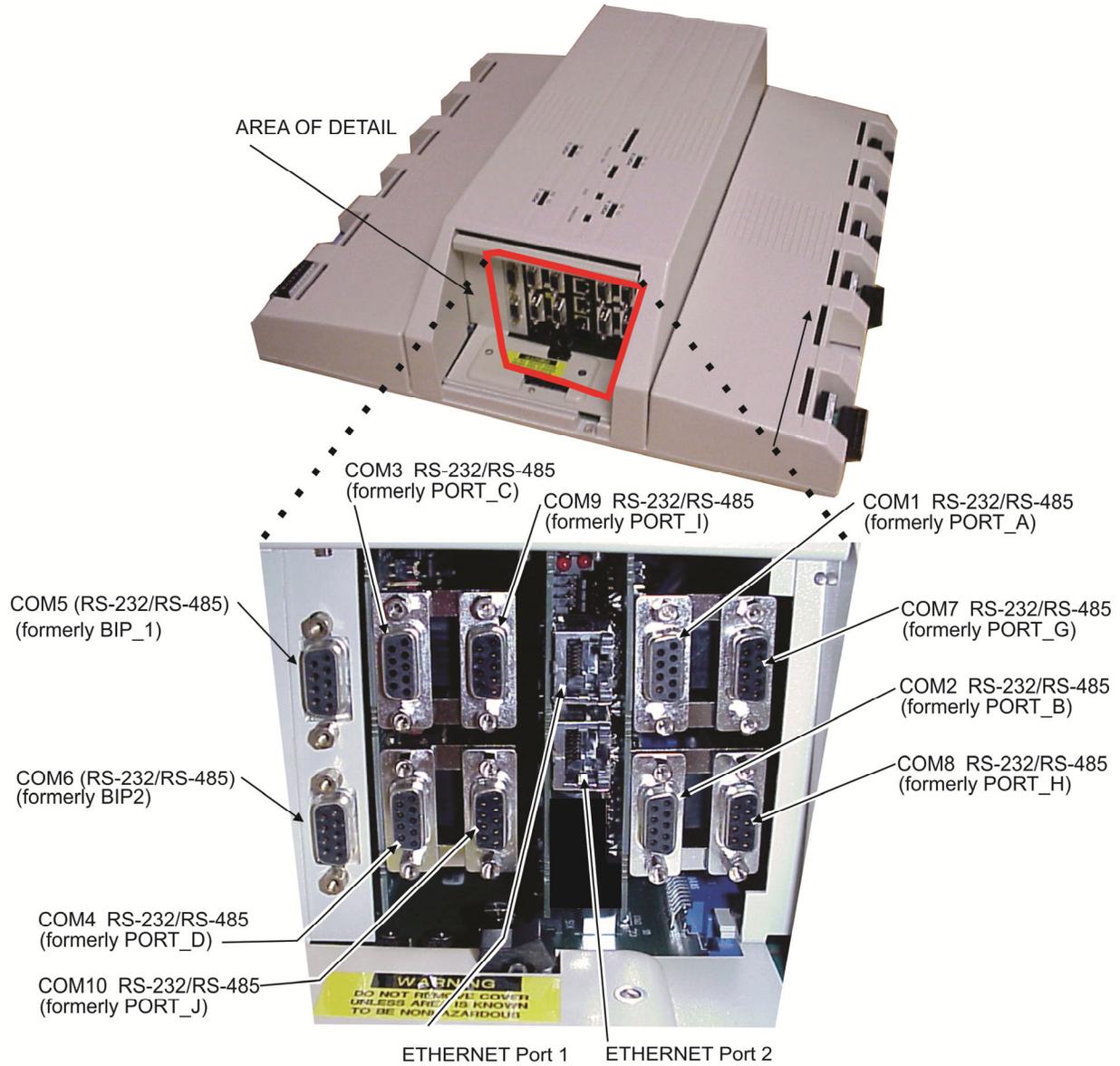
The ControlWave_10 (CW_10) is an RTU 3310 chassis and I/O upgraded with new ControlWave CPU and multi-function interface boards (MFIB).



CW_10

ControlWave CW_30

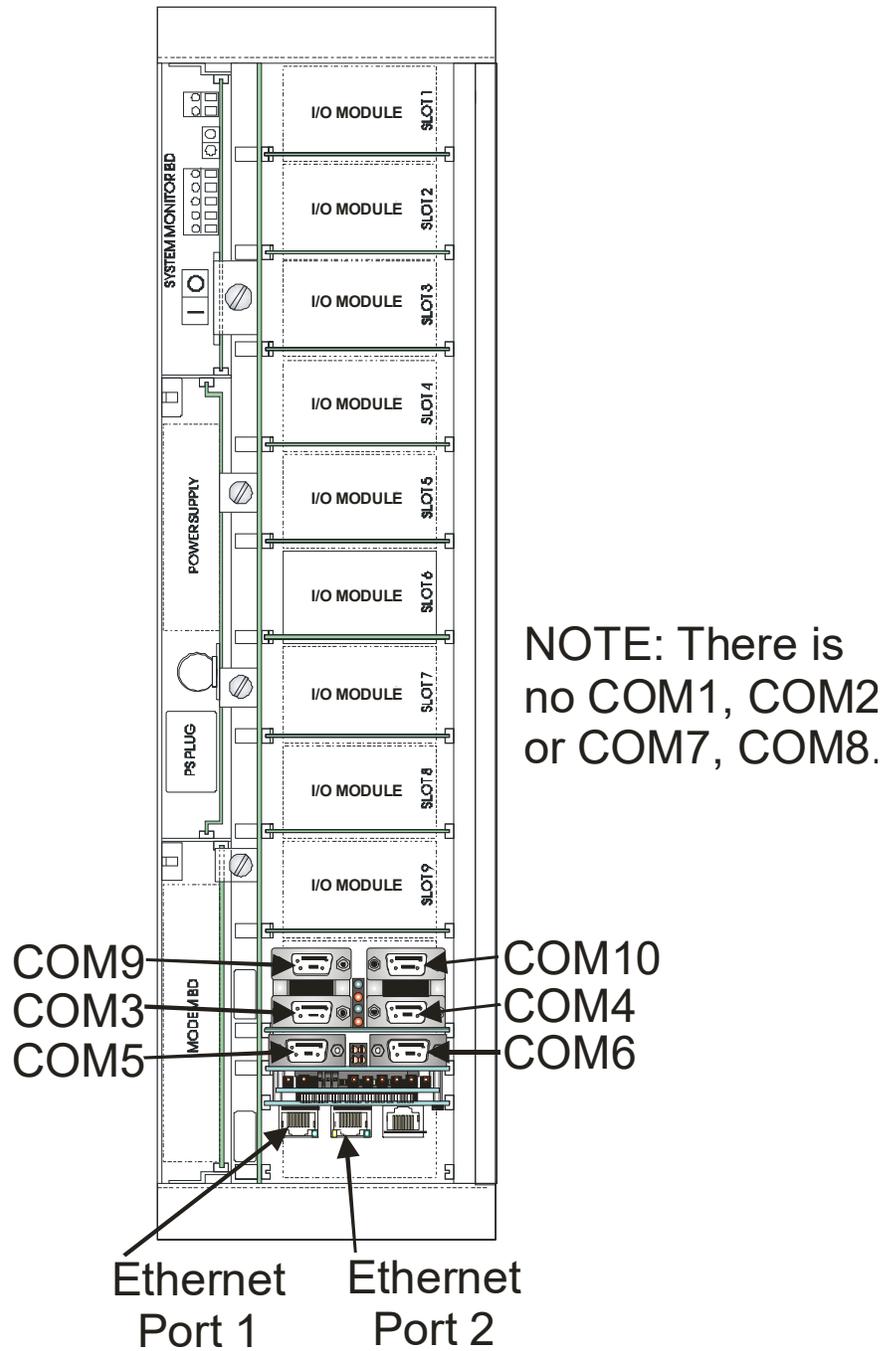
The ControlWave_30 (CW_30) is a DPC 3330 chassis and I/O upgraded with new ControlWave CPU and communication boards.



CW_30

ControlWave CW_35

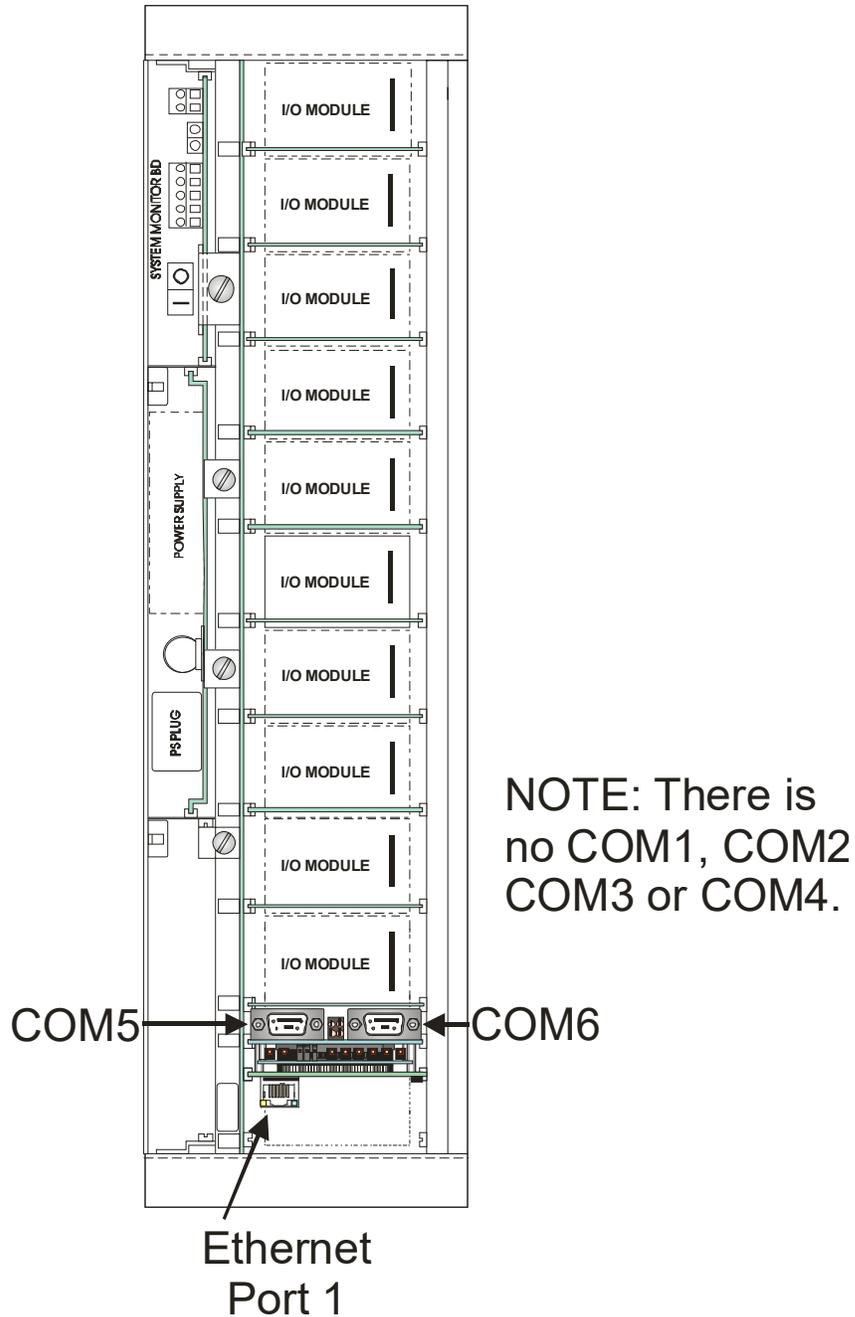
The ControlWave_35 (CW_35) is a DPC 3335 chassis and I/O upgraded with new ControlWave CPU and communication boards.



CW_35

ControlWave CW_31

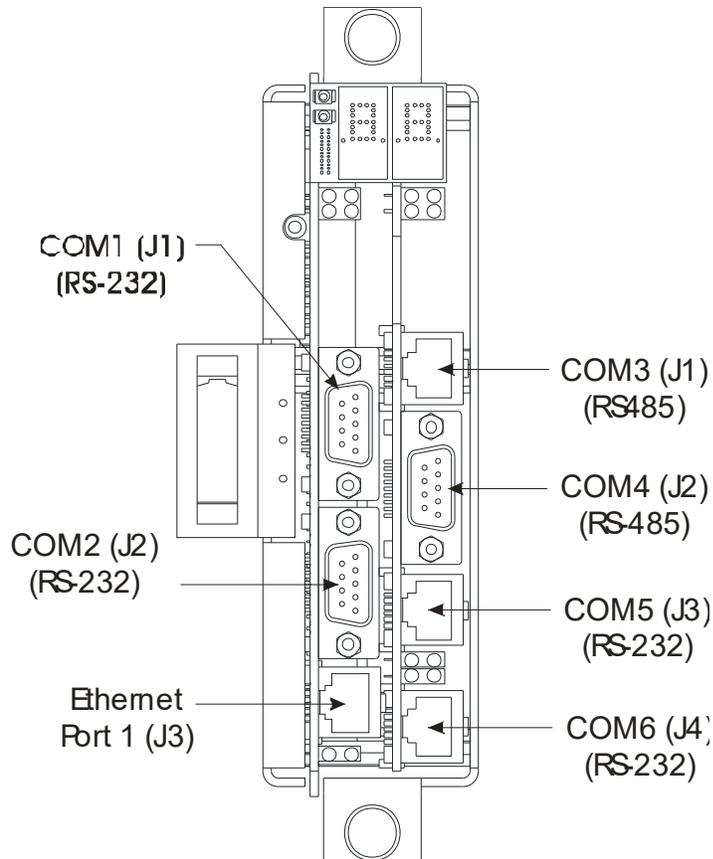
The ControlWave_31 (CW_31) is an RIO 3331 Remote I/O Rack chassis and I/O upgraded with new ControlWave communication boards.



CW_31

ControlWave I/O Expansion Rack

The ControlWave I/O Expansion Rack is a chassis containing additional I/O boards for a ControlWave host. It is also used as the shared I/O in certain redundancy configurations.



ControlWave I/O Expansion Rack

Methods for Communicating with ControlWave-series Controllers.

There are several ways to communicate with the ControlWave-series controllers including:

- Bristol Synchronous / Asynchronous Protocol (BSAP) - The ControlWave can be part of an OpenBSI BSAP network, and beginning with firmware release CWPO2, it can serve as a BSAP Master node.
- Internet Protocol (IP) - The ControlWave can be connected to an IP network of ControlWave or Network 3000 nodes. This can utilize Ethernet or serial Point-to-Point Protocol (PPP). Other IP options are available as well, e.g. Open Modbus. NOTE: Not all units include Ethernet ports.
- ControlWave Designer Protocol - The ControlWave has its own native protocol for communication with ControlWave Designer software. ControlWave Designer protocol can be transmitted via serial links, TCP/IP or OpenBSI.

- Serial Modbus - This industry standard protocol allows communication between a ControlWave controller configured for MODBUS, and another MODBUS device.

The table, below, summarizes the major communication options. NOTE: Different units of the ControlWave series support different numbers of ports.

Protocol	Reasons to Use	Which software is used?
Bristol Synchronous / Asynchronous Protocol (BSAP)	<p>Collect data from ControlWave using OpenBSI. Download the ControlWave project. View web pages. Update FLASH parameters and soft switches</p> <p>NOTE: Peer-to-peer communication with other ControlWave or Network 3000 controllers requires Client / Server function blocks (firmware 02.00 or newer). Client/Server function blocks are discussed in the ControlWave Designer on-line help.</p>	OpenBSI Utilities suite (NetView, DataView, Downloader, etc.)
Internet Protocol (IP)	<p>Collect data from ControlWave using OpenBSI. Download the ControlWave project. View web pages. IP MODBUS communication</p> <p>NOTE: Peer-to-peer communication with other ControlWave or Network 3000 controllers requires Client / Server function blocks (firmware 02.00 or newer). Client/Server function blocks are discussed in the ControlWave Designer on-line help.</p>	OpenBSI Utilities suite (NetView, DataView, Downloader, etc.)
ControlWave Designer Protocol (using either OpenBSI DLL, serial DLL or TCP/IP DLL)	<p>Run ControlWave Designer in on-line mode, which includes: Performing debugging operations. Downloading ControlWave project.</p>	ControlWave Designer
Very Small Aperture Terminal (VSAT)	Send data via satellite links.	ControlWave Designer, OpenBSI NetView
Allen-Bradley DF1 Master / Slave	Exchange data with other Allen-Bradley DF1 devices	ControlWave Designer CUSTOM function block. Port configured for DF1 within the Ports page of the Flash Configuration Utility.
DNP3	Industry-standard protocol for SCADA, etc.	<p>ControlWave Designer CUSTOM function block. Port configured for DNP3 within the Ports page of the Flash Configuration Utility.</p> <p>This protocol requires 4.40 (or newer) firmware. It is NOT supported for ControlWave LP.</p>
CIP	Allen-Bradley protocol.	ControlWave Designer CUSTOM

Protocol	Reasons to Use	Which software is used?
		function block. This protocol requires 4.40 (or newer) firmware. It is NOT supported for ControlWave LP.
Serial MODBUS	Exchange data with other MODBUS devices (including <i>another</i> ControlWave configured for MODBUS communication.)	ControlWave Designer CUSTOM function block. Port configured for MODBUS within the Ports page of the Flash Config. Utility.
HART	Exchange data with HART devices through HART Interface Board (HIB) or a serial port – ControlWave Micro / EFM only.	ControlWave Designer HART function block This protocol requires 5.00 (or newer) firmware.
Foundation Fieldbus	Exchange data with FFbus devices through ControlWave Foundation Fieldbus Interface	Fieldbus function block Field Interface Configurator software. This protocol requires 5.10 (or newer) firmware. .It is NOT supported for ControlWave LP.

How do I configure the Ports on the ControlWave?

The Flash Configuration Utility, available in LocalView and NetView, is used to configure the ports on any ControlWave unit, as well as for setting other parameters such as the BSAP local address. The Flash Configuration Utility is discussed in detail in Chapter 5 of the *OpenBSI Utilities Manual* (document# D5081).

What are the factory default settings for communication ports?

Factory Defaults for Ethernet Ports

Depending upon the type of ControlWave, there may be up to three Ethernet ports. Ethernet ports are pre-configured at the factory with initial IP addresses and masks, as follows:

ETH1 IP Address: 10.0.1.1 IP Mask: 255.255.255.0

ETH2 IP Address: 10.0.2.1 IP Mask: 255.255.255.0

ETH3 IP Address: 10.0.3.1 IP Mask: 255.255.255.0

Because each unit shipping from the factory will have these initially pre-programmed, you should only use these addresses for 'bench' testing and configuration. These addresses

must be changed before putting ControlWave units on an actual network, since an address conflict would exist as soon as the second ControlWave unit was placed online.

Factory Defaults for Serial Ports

Default configuration for ControlWave Serial ports is included in the table, below. These default settings are activated any time the default switch is in the OFF position:

- The default switch on the ControlWave Process Automation Controller, ControlWave I/O Expansion Rack and ControlWave XFC is (SW1-3).
- The default switch on the ControlWave LP Controller is (SW4-3).
- The default switch on the ControlWave MICRO Controller, EFM, GFC, GFC-CL, Express, Express PAC, ControlWave_10, ControlWave_30, ControlWave_35, and ControlWave_31 is (SW2-3).
- The ControlWave, ControlWave MICRO, ControlWave Redundant Controller, and EFM initially ship from the factory with serial COM port 1 set to BSAP at 115,200. Once the default switch is OFF however, a factory default of PPP at 115,200 applies.

More details on the factory default settings of communication ports are included in the hardware manual.

Factory Defaults for ControlWave, CW Redundant Controller Serial Ports

Name	Baud Rate	Bits Per Character	Stop Bits	Parity	Protocol	Cable	Notes
Serial Port COM1	115200	8	1	NONE	Serial IP (PPP)	RS232 null modem cable	Use IP address 1.1.1.1.
Serial Port COM2	9600	8	1	NONE	BSAP Slave/ControlWave Designer	RS232 null modem cable	
Serial Port COM3	9600	8	1	NONE	BSAP Slave/ControlWave Designer	Either RS232 null modem cable OR RS485 cable depending upon how ordered from factory. Cable connectors vary depending upon type ordered.	CHOICE OF RS232/RS485 MADE WHEN ORDERED FROM FACTORY. CANNOT BE CHANGED IN THE FIELD.
Serial Port COM4	9600	8	1	NONE	BSAP Slave/ControlWave Designer	Either RS232 null modem cable OR RS485 cable depending upon how ordered from factory.	CHOICE OF RS232/RS485 MADE WHEN ORDERED FROM FACTORY. CANNOT BE CHANGED IN FIELD.

Factory Defaults for ControlWave I/O Expansion Rack Serial Ports

Name	Baud Rate	Bits Per Character	Stop Bits	Parity	Protocol	Cable	Notes
Serial Port COM1	9600	8	1	NONE	Serial IP (PPP)	RS232 null modem cable	Use IP address 1.1.1.1.
Serial Port COM2	9600	8	1	NONE	BSAP Slave/ControlWave Designer	RS-232 cable	
Serial Port COM3	9600	8	1	NONE	BSAP Slave/ControlWave Designer	RS485 cable	
Serial Port COM4	9600	8	1	NONE	BSAP Slave/ControlWave Designer	RS485 cable	
Serial Port COM5	9600	8	1	NONE	BSAP Slave/ControlWave Designer	RS232 cable	
Serial Port COM6	9600	8	1	NONE	BSAP Slave/ControlWave Designer	RS232 cable	

Factory Defaults for ControlWave MICRO, ControlWave EFM Serial Ports

Name	Baud Rate	Bits Per Character	Stop Bits	Parity	Protocol	Cable / Interface	Notes
Serial Port COM1	115200	8	1	NONE	Serial IP (PPP)	RS232 null modem cable	Use IP address 1.1.1.1. See Note below
Serial Port COM2	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable	
Serial Port COM3	9600	8	1	NONE	BSAP Slave/ControlWave Designer	RS485 cable	Configured via CPU switch SW3.
Serial Port COM4	9600	8	1	NONE	BSAP Slave/ControlWave Designer	RS232 null modem cable	OPTIONAL – requires ECOM Board in chassis slot 3.
Serial Port COM5	9600	8	1	NONE	BSAP Slave/ControlWave Designer	RS485 cable	OPTIONAL – requires ECOM Board in chassis slot 3. Configured by

Name	Baud Rate	Bits Per Character	Stop Bits	Parity	Protocol	Cable / Interface	Notes
							switch SW1 on ECOM board.
Serial Port COM6	9600	8	1	NONE	BSAP Slave/ControlWave Designer	Requires coaxial RF cable to connect MDS / FreeWave® spread spectrum modem (radio).	OPTIONAL – requires ECOM Board in chassis slot 3. See 2.3.3.5 in CI-ControlWave MICRO for radio installation steps.
Serial Port COM7	9600	8	1	NONE	BSAP Slave/ControlWave Designer	RJ11 connector for connecting to 56K PSTN modem	OPTIONAL – requires ECOM Board in chassis slot 3.
Serial Port COM8	9600	8	1	NONE	BSAP Slave/ControlWave Designer	RS232 null modem cable	OPTIONAL – requires ECOM Board in chassis slot 4.
Serial Port COM9	9600	8	1	NONE	BSAP Slave/ControlWave Designer	RS485 cable	OPTIONAL – requires ECOM Board in chassis slot 4. Configured by switch SW1 on ECOM board
Serial Port COM10	9600	8	1	NONE	BSAP Slave/ControlWave Designer	Requires coaxial RF cable to connect MDS / FreeWave® spread spectrum modem (radio).	OPTIONAL – requires ECOM Board in chassis slot 4. See 2.3.3.5 in CI-ControlWave MICRO for radio installation steps.
Serial Port COM11	9600	8	1	NONE	BSAP Slave/ControlWave Designer	RJ11 connector for connecting to 56K PSTN modem	OPTIONAL – requires ECOM Board in chassis slot 4.

Factory Defaults for ControlWave MICRO I/O Expansion Rack Serial Ports

Name	Baud Rate	Bits Per Character	Stop Bits	Parity	Protocol	Cable / Interface	Notes
Serial	9600	8	1	NONE	Gould MODBUS	RS232 null	

Name	Baud Rate	Bits Per Character	Stop Bits	Parity	Protocol	Cable / Interface	Notes
Port COM1					Slave – RTU mode.	modem cable	
Serial Port COM2	115200	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable	
Serial Port COM3	115200	8	1	NONE	BSAP Slave/ ControlWave Designer	RS485 cable	Configured via CPU switch SW3.
Serial Port COM4	115200	8	1	NONE	BSAP Slave/ ControlWave Designer	RS232 null modem cable	OPTIONAL – requires ECOM Board in chassis slot 3.
Serial Port COM5	115200	8	1	NONE	BSAP Slave/ ControlWave Designer	RS485 cable	OPTIONAL – requires ECOM Board in chassis slot 3. Configured by switch SW1 on ECOM board.
Serial Port COM6	115200	8	1	NONE	BSAP Slave/ ControlWave Designer	Requires coaxial RF cable to connect MDS / FreeWave® spread spectrum modem (radio).	OPTIONAL – requires ECOM Board in chassis slot 3. See 2.3.3.5 in CI-ControlWave MICRO for radio installation steps.
Serial Port COM7	115200	8	1	NONE	BSAP Slave/ ControlWave Designer	RJ11 connector for connecting to 56K PSTN modem	OPTIONAL – requires ECOM Board in chassis slot 3.
Serial Port COM8	115200	8	1	NONE	BSAP Slave/ ControlWave Designer	RS232 null modem cable	OPTIONAL – requires ECOM Board in chassis slot 4.
Serial Port COM9	115200	8	1	NONE	BSAP Slave/ ControlWave Designer	RS485 cable	OPTIONAL – requires ECOM Board in chassis slot 4. Configured by switch SW1 on ECOM board
Serial Port COM10	115200	8	1	NONE	BSAP Slave/ ControlWave Designer	Requires coaxial RF cable to connect MDS / FreeWave®	OPTIONAL – requires ECOM Board in chassis slot 4.

Name	Baud Rate	Bits Per Character	Stop Bits	Parity	Protocol	Cable / Interface	Notes
						spread spectrum modem (radio).	See 2.3.3.5 in CI-ControlWave MICRO for radio installation steps.
Serial Port COM11	115200	8	1	NONE	BSAP Slave/ControlWave Designer	RJ11 connector for connecting to 56K PSTN modem	OPTIONAL – requires ECOM Board in chassis slot 4.

Factory Defaults for ControlWave Gas Flow Computer Classic (GFC-CL) Serial Ports

Name	Baud Rate	Bits Per Character	Stop Bits	Parity	Protocol	Cable / Interface	Notes
Serial Port COM1	115200	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable	Can also serve as Local Port. Different connectors are available depending on usage.
Serial Port COM2	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485 cable	Can serve as either RS232 or RS485. If using RS485, switch SW4 used for configuration. This port supports radio / modem option.
Serial Port COM3	9600	8	1	NONE	BSAP Slave/ControlWave Designer	RS485 cable	Configured via CPU switch SW3.

Factory Defaults for ControlWave Gas Flow Computer (GFC) / ControlWave Express / ControlWave Express PAC Serial Ports

Name	Baud Rate	Bits Per Character	Stop Bits	Parity	Protocol	Cable / Interface	Notes
Serial Port COM1	115200	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable	Can also serve as Local Port. Different connectors are available depending on usage.
Serial Port COM2	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable	This port supports radio / modem option.

Name	Baud Rate	Bits Per Character	Stop Bits	Parity	Protocol	Cable / Interface	Notes
Serial Port COM3	9600	8	1	NONE	BSAP Slave/ControlWave Designer	RS232 null modem cable or RS485 cable	Can serve as either RS232 or RS485. If using RS485, configured via CPU switch SW3.

Factory Defaults for ControlWave Gas Flow Computer (XFC) Serial Ports

Name	Baud Rate	Bits Per Character	Stop Bits	Parity	Protocol	Cable / Interface	Notes
Serial Port COM1	115200	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable	Local Port
Serial Port COM2	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable	Network Port
Serial Port COM3	9600	8	1	NONE	BSAP Master/ControlWave Designer	RS485 cable	Configured as BSAP Master for communication with 3808 MVT Transmitter.

Factory Defaults for ControlWave_10 (CW_10) Serial Ports

Name	Baud Rate	Bits Per Character	Stop Bits	Parity	Protocol	Cable / Interface	Notes
Serial Port COM1	115200	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable	Configured by jumper W4 on CMFIB. On original RTU 3310 known as Port A.
Serial Port COM2	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Configured by jumpers W5 and W7 and switch SW2 on CMFIB. On original RTU 3310 known as Port B.
Serial Port COM3	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Configured by jumpers W8 through W14 and switch SW3 on CMFIB. On original RTU 3310 known as Port C.
Serial Port	9600	8	1	NONE	BSAP Slave / ControlWave	RS232 null modem cable	Configured by jumpers W15

Name	Baud Rate	Bits Per Character	Stop Bits	Parity	Protocol	Cable / Interface	Notes
COM4					Designer	or RS485	through W21 and switch SW4 on CMFIB. On original RTU3310 known as Port D.
Serial Port COM5	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Configured by jumpers W9 through W15 and switch SW3 on CCPU board. On original RTU 3310 known as BIP1.
Serial Port COM6	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Configured by jumpers W16 through W22 and switch SW4 on CCPU board. On original RTU 3310 known as BIP2.

Factory Defaults for ControlWave_30 (CW_30) Serial Ports

Name	Baud Rate	Bits Per Character	Stop Bits	Parity	Protocol	Cable / Interface	Notes
Serial Port COM1	115200	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Port 1 on the <i>first</i> CCB. Configured by jumpers W2 through W8 and switch SW1 on CCB. On original DPC 3330 known as Port A.
Serial Port COM2	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Port 2 on the <i>first</i> CCB. Configured by jumpers W9 through W16 and switch SW3 on CCB. On original DPC 3330 known as Port B.
Serial Port COM3	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Port 3 on the <i>second</i> CCB. Configured by jumpers W2 through W8 and switch SW1 on CCB. On original DPC

Name	Baud Rate	Bits Per Character	Stop Bits	Parity	Protocol	Cable / Interface	Notes
							3330 known as Port C.
Serial Port COM4	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Port 4 on the <i>second</i> CCB. Configured by jumpers W9 through W16 and switch SW3 on CCB. On original DPC 3330 known as Port D.
Serial Port COM5	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Configured by jumpers W9 through W15 and switch SW3 on CCPU board. On original DPC 3330 known as BIP1.
Serial Port COM6	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Configured by jumpers W16 through W22 and switch SW4 on CCPU board. On original DPC 3330 known as BIP2.
Serial Port COM7	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Port 7 on the <i>first</i> CCB. Configured by jumpers W17 through W23 and switch SW2 on CCB. On original DPC 3330 known as Port G.
Serial Port COM8	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Port 8 on the <i>first</i> CCB. Configured by jumpers W24 through W30 and switch SW4 on CCB. On original DPC 3330 known as Port H.
Serial Port COM9	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Port 9 on the <i>second</i> CCB. Configured by jumpers W17 through W23 and switch SW2 on CCB. On original DPC 3330 known as Port

Name	Baud Rate	Bits Per Character	Stop Bits	Parity	Protocol	Cable / Interface	Notes
							I.
Serial Port COM10	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Port 10 on the <i>second</i> CCB. Configured by jumpers W24 through W30 and switch SW4 on CCB. On original DPC 3330 known as Port J.

Factory Defaults for ControlWave_35 (CW_35) Serial Ports

Note: Because the CW_35 does NOT support a communications board in Slot 13, and for firmware compatibility purposes, the following ports do NOT exist on these units: COM1, COM2, COM7, and COM8.

Name	Baud Rate	Bits Per Character	Stop Bits	Parity	Protocol	Cable / Interface	Notes
Serial Port COM3	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Port 3 on the CCB. Configured by jumpers W2 through W8 and switch SW1 on CCB. On original DPC 3335 known as Port C.
Serial Port COM4	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Port 4 on the CCB. Configured by jumpers W9 through W15 and switch SW3 on CCB. On original DPC 3335 known as Port D.
Serial Port COM5	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Port on the CCPU. Configured by jumpers W10 through W16 and switch SW3 on CCPU. On original DPC 3335 known as Port BIP1.
Serial Port COM6	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Port on the CCPU. Configured by jumpers W18

Name	Baud Rate	Bits Per Character	Stop Bits	Parity	Protocol	Cable / Interface	Notes
							through W24 and switch SW4 on CCPU. On original DPC 3330 known as Port BIP2.
Serial Port COM9	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Port 9 on the CCB. Configured by jumpers W16 through W23 and switch SW2 on CCB. On original DPC 3335 known as Port I.
Serial Port COM10	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Port 10 on the CCB. Configured by jumpers W24 through W30 and switch SW4 on CCB. On original DPC 3335 known as Port J.

Factory Defaults for ControlWave_31 (CW_31) Serial Ports

Note: Because the CW_31 does NOT support a communications board in Slot 13 or Slot 10, and for firmware compatibility purposes, the following ports do NOT exist on these units: COM1, COM2, COM3, COM4, COM7, COM8, COM9, COM10.

Name	Baud Rate	Bits Per Character	Stop Bits	Parity	Protocol	Cable / Interface	Notes
Serial Port COM5	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Port on the CCPU. Configured by jumpers W10 through W16 and switch SW3 on CCPU. On original RIO 3331 known as Port BIP1.
Serial Port COM6	9600	8	1	NONE	BSAP Slave / ControlWave Designer	RS232 null modem cable or RS485	Port on the CCPU. Configured by jumpers W18 through W24 and switch SW4 on CCPU. On original RIO 3331 known as Port BIP2.

How can the port configuration be changed?

If you want to use port settings other than the defaults, you must connect to the ControlWave unit, and change the port configuration settings. The new settings are saved in the 512K FLASH BIOS of the ControlWave unit. Port configurations may be changed via the Flash Configuration Utility.

Important

When you make changes to port configurations, some of the changes will take effect immediately after you exit the Flash Configuration Utility. Other changes will not take effect until after you have reset the ControlWave unit (turned it off and then back on).

Dialing - An Overview

Beginning with ControlWave firmware version 04.00, dialing is supported via the DIAL_CTRL function block. To configure dialing, you must:

- Configure the communication port from the 'Ports' page of the Flash Configuration utility.
- Identify the communication port used for dialing via the `_Px_DIAL_PORT` system variable, in your ControlWave project. *Dialing will NOT function unless the port is identified as a dial port.* System variables are configured in the System Variable Wizard.
- Configure your ControlWave project to handle dialing. Use the DIAL_CTRL function block to specify the various dialing parameters. See the on-line help in ControlWave Designer for more information. Any RS-232 port can be a dial port, and you can have multiple dial ports and multiple DIAL_CTRL function blocks in your project, but each port must have a dedicated modem.
- Configure the external modem which will perform the dialing. Currently, we offer the MultiTech® Systems Embedded Data FAX Modem. See the documentation accompanying this modem for configuration details.

Serial Port Sharing between the BSAP Slave and Custom Slave Protocols:

Normally, when you configure a serial port on a ControlWave-series controller, the port only uses a single protocol, such as BSAP or Modbus and that protocol has full control over that port. Under certain circumstances, though, you can configure serial port sharing which allows a single port to serve as both a BSAP slave, and as a custom slave using a custom protocol.

To implement serial port sharing, you must follow these rules:

1. Define the serial port as the appropriate custom slave port, such as Modbus slave, Allen-Bradley slave, etc.
2. Ensure that the port characteristics, e.g. baud rate, start/stop bits, and parity, used by both the BSAP master, and the custom master match exactly.
3. When switching from one master to the other master, the first communication protocol must relinquish control over the port and allow the second protocol to establish successful communication with the new master using one of the following methods:
 - a. The communication line must be quiet for approximately 1 minute or
 - b. The new master must make approximately five attempts to establish communications with the slave, even if it receives no response.
 - c. You can set the `_Pn_INH_BSAP_SLAVE` system variable to TRUE (firmware 05.43 and newer) to inhibit the serial port from accepting any BSAP slave communications.

Serial port sharing has been tested for switching between the BSAP and Modbus slaves as well as between BSAP and Allen-Bradley DF1 slaves. For any other protocol combinations, you must verify that serial port sharing works properly for your application.

Compiling

When you have finished editing your ControlWave Designer project, it must be compiled. The compilation process takes your project (programs, function blocks, tasks, etc.) and generates machine-readable code from it. After successful compilation, the machine readable code for the project can be **downloaded** into the ControlWave-series controller or the I/O Simulator.

Note: If a particular POU has NOT been compiled, its name will have an asterisk (*) next to it in the project tree.

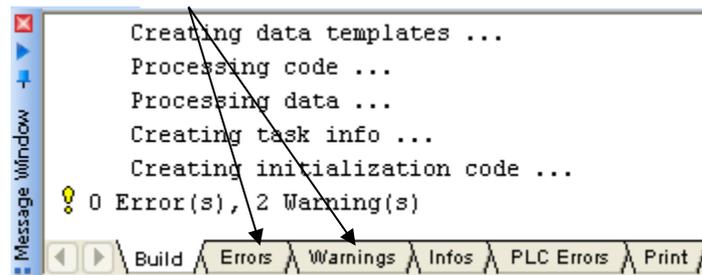
The compilation process checks for any syntactical errors in your project, and also issues warnings about possible problems with the structure of the project. It does NOT check for logic errors in your control strategy, however.



To compile the project, click on the icon shown at left, or go to the menu bar, and click as follows: Build→Make

Various messages will appear on the screen.

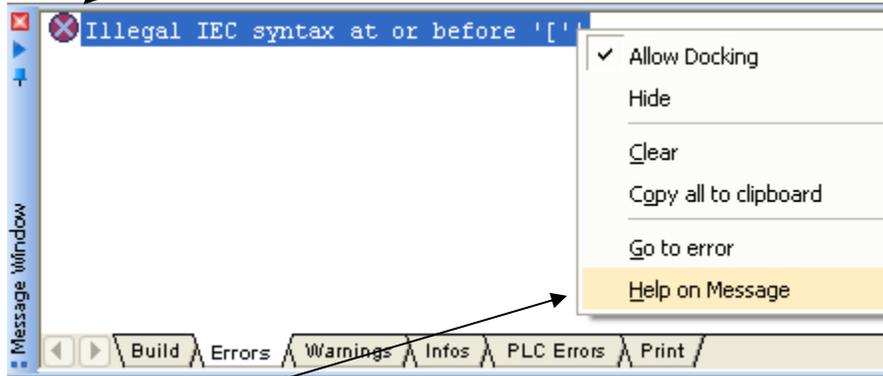
If there are errors or warnings, click on the “Errors” or “Warnings” tab for more information.



If there are errors or warnings generated during the compilation, you can view them by clicking on the 'Errors' or 'Warnings' tabs, respectively. Often, you can double-click on the error listed in the error window, and its location in the project will be identified.

For more information about what a particular error message means, *right-click* on the error message, then choose **“Help on Message”** from the pop-up menu (if it is available.)

Right-click on the error message to jump to the location in the file where the compiler found this error.



Double-click on the error message and choose “Help on Message” from the pop-up menu to get more information on the error.

Conditional Logic

Certain function blocks (for example, LIST function blocks) should only be executed once at application cold start *and* at application warm start.

Note

Lists can also be created using the DB_LOAD function block (without any LIST function blocks). See the Variable Extension Wizard for more information.

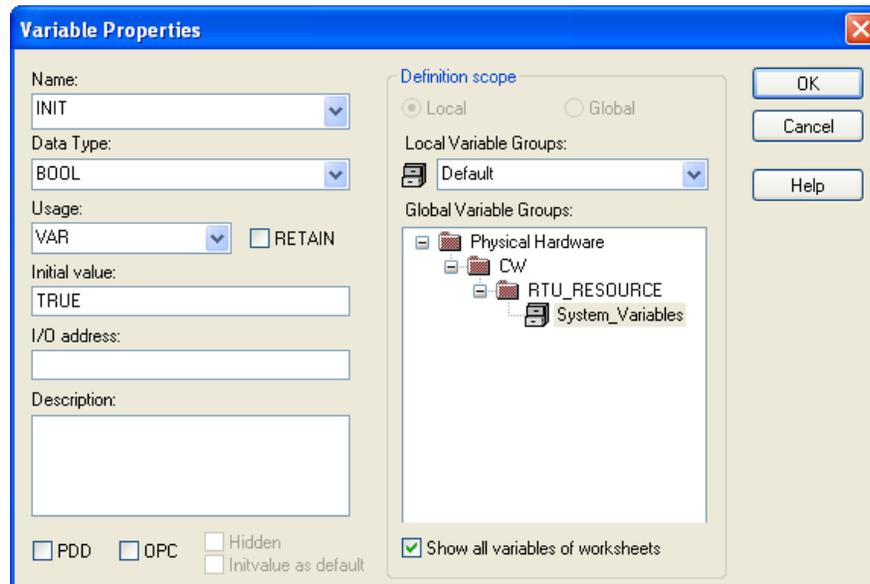
If you attempt to execute these function blocks multiple times, the system tries to define the list structures multiple times, which results in an error code, each time the LIST is executed. While this does **not** prevent your ControlWave project from executing properly, it can burden you with unnecessary error messages, which could prevent you from seeing more useful error messages.

To avoid this, you can use conditional logic to execute the list only once.

Using Conditional Logic to Execute a Function Block Only Once

This is most easily accomplished when using a POU written in Structured Text (ST) language.

In the example, below, a variable called INIT, of type BOOL has been created, which has an initial state of TRUE. Note that for this type of operation, the variable should NOT be marked as "RETAIN", because that would prevent the list from being defined following an application warm start.



At application start up, the INIT variable is read as part of an IF - ENDIF block. Since it is TRUE, the LIST function block will be executed, thereby defining the list. At the very end of

the IF-ENDIF block, the INIT variable is set to FALSE, to prevent that block of code from executing again:

ST CODE EXCERPT:

IF (INIT) THEN

_LIST_10_1.iListNumber := 1;

_LIST_10_1.iAnyElement1 := LOAD_NAME;

_LIST_10_1.iAnyElement2 := FUN1_TEST_CNT;

_LIST_10_1.iAnyElement3 := FUN_TEST1_HLT;

_LIST_10_1.iAnyElement4 := FUN_OUT_001;

_LIST_10_1.iAnyElement5 := FUN_ERROR1_CNT;

_LIST_10_1.iAnyElement6 := FUN2_TEST_CNT;

_LIST_10_1.iAnyElement7 := FUN_TEST2_HLT;

_LIST_10_1();

INIT:= FALSE;

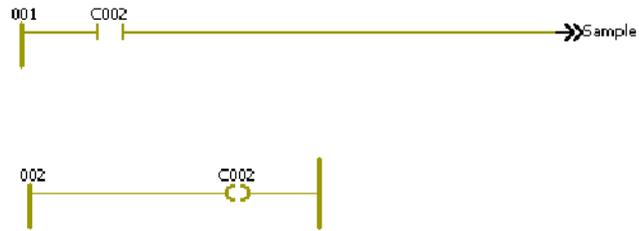
END_IF;

Using LD:

To execute a block of code only once in Ladder Language (LD), you can use the '>>' jump statement to effectively skip the code on subsequent executions. Execution proceeds left-to-right and top-to-bottom.

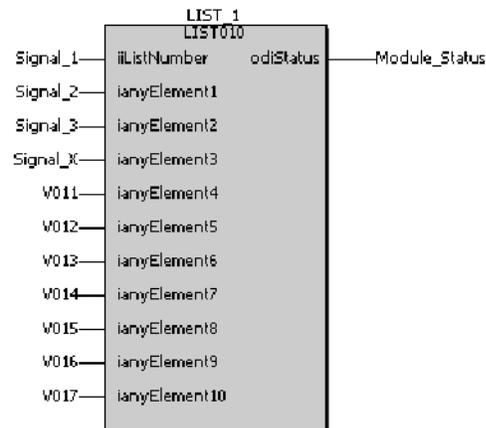
In the first line, the jump statement at the far left is ignored, because 'C002' is FALSE, and so execution continues on the next line

The second line turns on 'C002'.



The third line (LIST function block) is executed.

On subsequent executions, the second and third lines will be skipped, because 'C002' is now TRUE -- program execution will jump to the 'Sample' label, where additional code could be executed.



Sample:
(*Additional code could follow the 'Sample' label.*)

This will continue on all subsequent executions (until the system is restarted), because the jump condition is satisfied.

DataView

You can use OpenBSI's DataView to call up the value of variables in your ControlWave project. For this work, OpenBSI communications (NetView or LocalView) must already be communicating with the ControlWave.

Note:

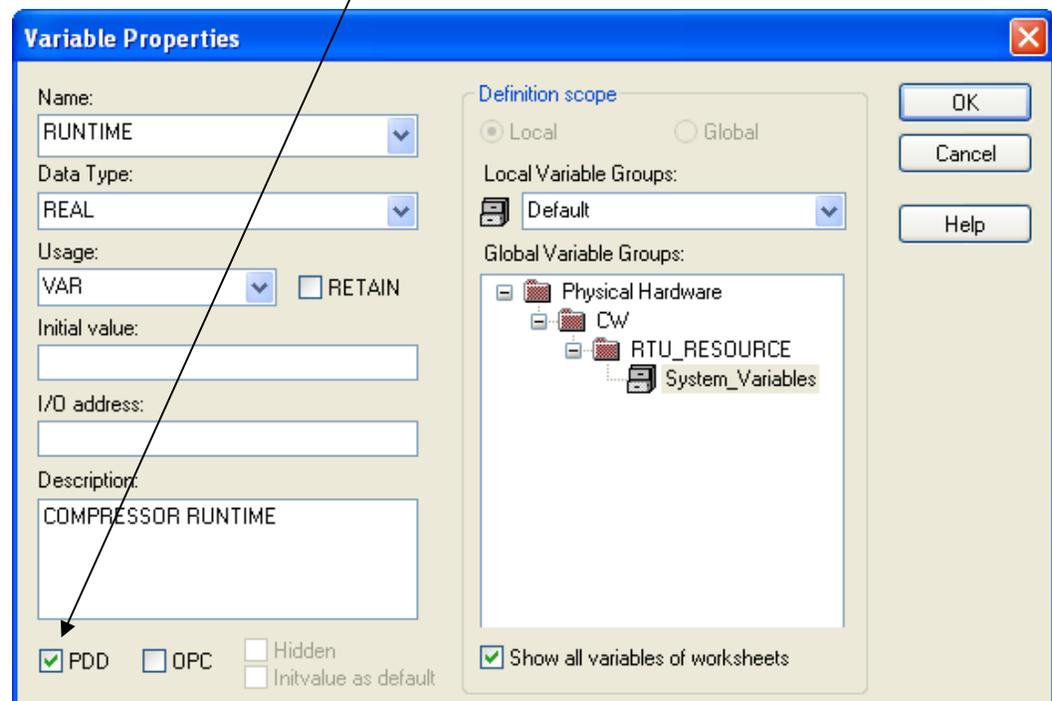
For a full discussion of how to use DataView, please refer to *Chapter 8* of the *OpenBSI Utilities Manual* (part number D301414X012).

Before you begin:

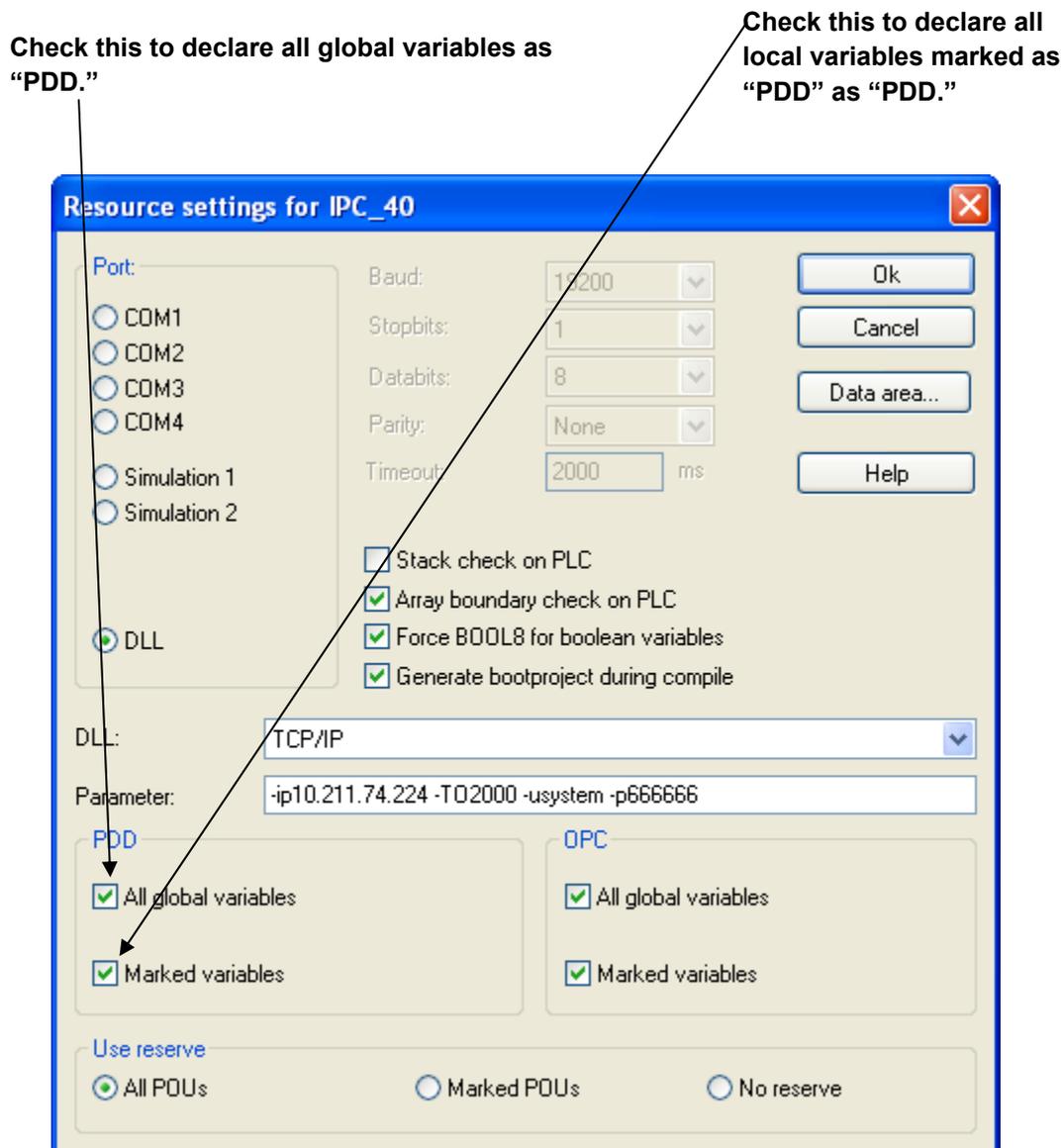
Note:

Only global variables (typically I/O global variables), or variables which have been marked as "PDD" in ControlWave Designer will be visible in DataView.

Click here to mark the variable as "PDD."



In addition, you must have made the proper settings for PDD in the Resource Settings dialog box (shown below):



Calling up ControlWave Data in DataView

Step 1. In NetView or LocalView, *right-click* on the icon of the ControlWave you want to access, and choose **RTU → DataView** from the pop-up menus.

Step 2. Sign on.

Step 3. Click on the Signal Search icon  or click on **File → New**, and then click on **"Signal Search"** in the New list box. Either method will call up the Signal Search Properties dialog box.

Step 4. The Signal Search dialog box should appear, with the ControlWave's node name in the **"Node"** field.

To see ALL variables which are global, or have been marked as PDD, just click on **[OK]**.

To search for a specific variable, follow the instructions in *Chapter 8* of the *OpenBSI Utilities Manual* (part number D301414X012).

Debugging – An Overview

Once you have corrected all syntactical errors and have successfully compiled your project, you can download it into the ControlWave-series controller or the I/O Simulator (see *Downloading* for information on how to do this).

ControlWave Designer supports several different debugging techniques to help isolate logic problems you find in your running ControlWave project. The techniques supported include:

- Using the Watch Window to display only a certain set of variables which you are interested in.
- Using the Cross-Reference Window to display where all variables and function blocks are used within a project.
- Using the Patch POU feature to perform edits to a running project, without stopping execution.
- Using breakpoints to stop execution at particular points in a POU, allowing the user to step through code, and view the results of execution at each step.
- Using the Force/Overwrite options to manually change values of variables in the running project.

Important

We recommend that debugging be performed only in the I/O Simulator, or in a ControlWave-series controller that is NOT currently connected to a running plant or process. This is because debug operations such as setting breakpoints, or forcing variables could cause an upset to a critical process.

These techniques are all performed in on-line Debug Mode, when you are communicating on-line with the ControlWave-series controller, or the I/O Simulator, from within ControlWave Designer.

Starting Debug Mode

With the project executing in the ControlWave, or the I/O Simulator, you can put ControlWave Designer into on-line Debug Mode to view the current values of variables, or perform debugging operations. To enter Debug Mode, click on the 'Debug On/Off' icon, shown above, or click on **Online**→**Debug**.

Using the Watch Window

The Watch Window allows you to view just the variables you are interested in. To open the Watch Window, click on **View**→**Watch Window**.

Using the Cross-Reference Window

The Cross-Reference Window displays in detail, how and where any particular variable, function block, etc. is used within the project.

To open the Cross-Reference Window, click on **Build → Build Cross References**.

Right-click in the Cross-References Window, and choose **“Build Cross References”** from the pop-up menu.

The Cross Reference tables will be generated and displayed in the window. You may want to drag the window borders to display additional information.

Once the cross-reference information is displayed, you can double-click on any variable name, and the worksheet containing that particular usage of the variable will be displayed.

The screenshot shows the Cross-Reference Window with the following table:

Variable	POU/Worksheet	Access	Comm...
BEGIN_TEST	MAIN_MUL2.MAIN_MUL2	Read	
BEGIN_TEST	STARTUP.STARTUP	Read	
BEGIN_TEST	STARTUP.STARTUP	Write	:=
FLA_IMM2_done	MAIN1.MAIN1	Read	AND
Hi_Alarm_Limit	ALARMS.ALARMSS	Read	
Hi_Hi_Alarm	ALARMS.ALARMSS	Write	:=
Hi_Hi_Alarm	ALARMS.ALARMSS	Read	+
Hi_Hi_Alarm	ALARMS.ALARMSS	Read	>
Hi_Hi_Alarm	ALARMS.ALARMSS	Write	:=
Hi_Hi_Alarm_Limit	ALARMS.ALARMSS	Read	

The callout box shows the following code snippet:

```

2.6250000E+001 Hi_Hi_Alarm := Hi_Hi_Alarm + 1.2
2.6250000E+001 If (Hi_Hi_Alarm > Hi_Hi_Alarm_Li
2.6250000E+001   Hi_Hi_Alarm :=0.0;
3.2000000E+001 End_If;
3.2000000E+001 High_Alarm := High_Alarm + 1.0;
3.2000000E+001 If (High_Alarm > Hi_Alarm_Limit)
3.2000000E+001   High_Alarm:=0.0;
0.0000000E+000 End_If;
0.0000000E+000 Low_Low_Alarm := Low_Low_Alarm;
0.0000000E+000 If (Low_Low_Alarm < Lo_Lo_Alarm
0.0000000E+000   Low_Low_Alarm := 0.0;
0.0000000E+000 End_If;
  
```

Double-click on a variable name in the cross-reference window, and the worksheet containing that usage of the variable opens.

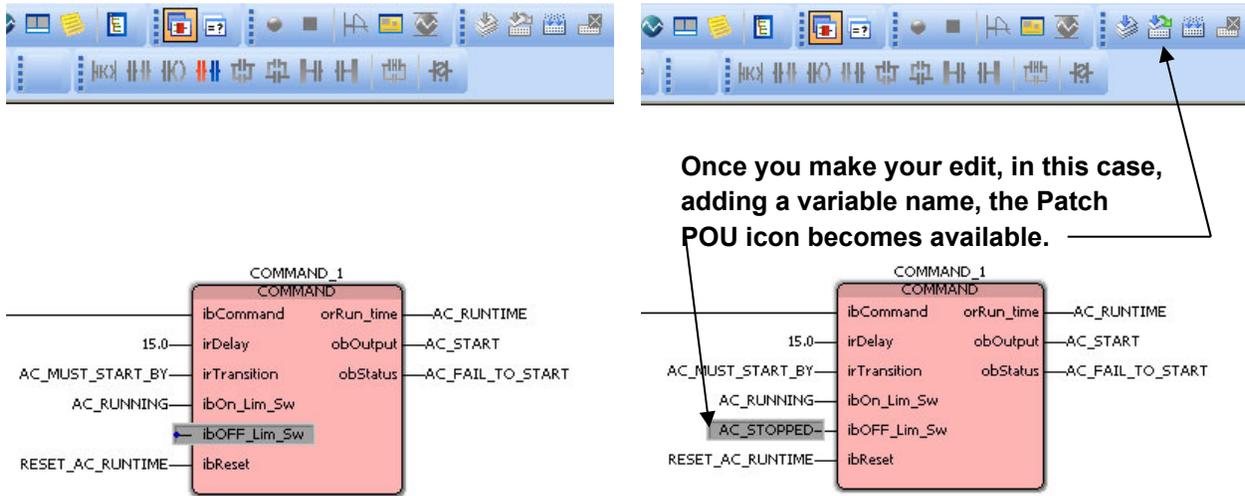
On-line Editing with Patch POU

On-line editing allows you to make changes to the ControlWave project running in the controller, or in the I/O Simulator, without pausing the execution of the project.

First, clear all breakpoints. (Clearing breakpoints is discussed later in this section.)

If you are already in on-line Debug Mode, you must click on the  button to exit Debug Mode, or click on “Online→Debug”.

Now make the edits you want to make in your POU. For example, the COMMAND function block shown on the left, below, has no variable for the `ibOFF_Lim_Sw` parameter defined. On the right we have added it. Because we have made an off-line edit, the ‘Patch POU’ icon now becomes available.



Click on the ‘Patch POU’ icon  and the edits you have made to your POU will be compiled, and downloaded into the running project, without stopping execution.

On-line Debug Mode will automatically be started so you can observe the behavior of the newly modified POU in your running project.

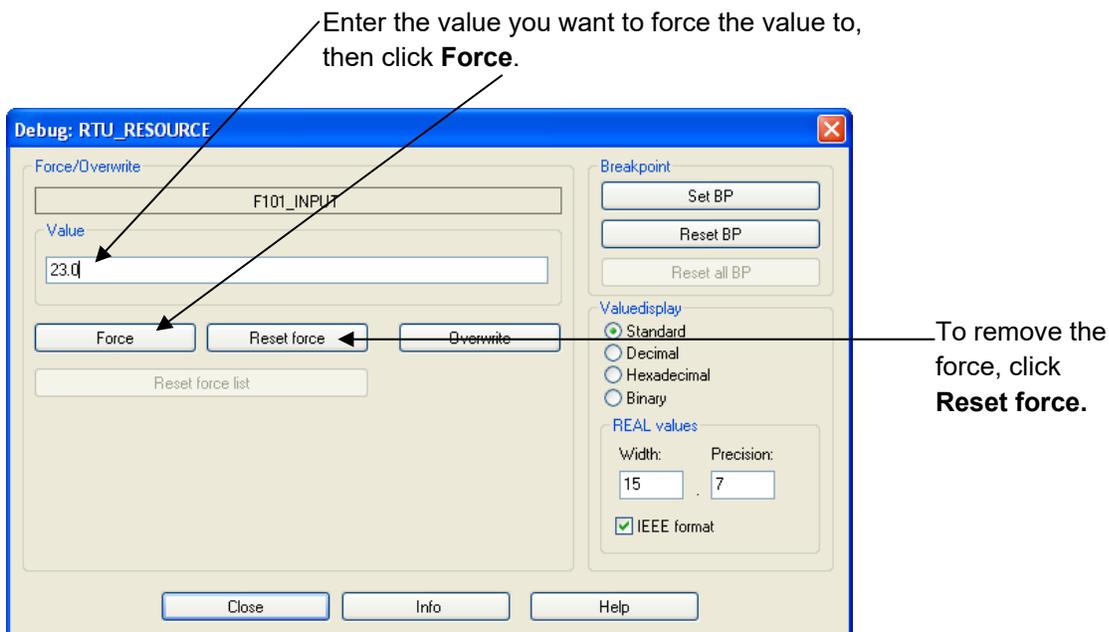
Using the Force/Overwrite Options

Force and Overwrite are ways to manually change the values of variables in the running ControlWave project.

Forcing an I/O Variable's Value

Force should only be used with I/O variables. To **force** an I/O variable changes its value at the point it is sent to the I/O board. Other logic within POUs can alter the I/O variable's value within the project, but at the point it reaches the I/O, the force action will be applied. That I/O point will stay at its forced value, until changed by the user, or until the force is removed.

- In Debug Mode, go into the worksheet for one of your POUs, for example, one of the graphical worksheets, and find the I/O variable you want to force.
- Double-click on the I/O variable that you want to force. The Debug: RTU Resource dialog box will appear.
- Select the new value for the forced variable. (For REAL variables, enter a value in the **"Value"** field; for BOOL variables, select either **"TRUE"** or **"FALSE"**. After specifying the value, click on the **[Force]** button, and the force will be applied at the I/O point.
- To stop applying the forced value, double-click on the I/O variable, and click on the **[Reset force]** button in the Debug: RTU Resource dialog box.



Note:

The force operation is only applied at the I/O point, as the value is sent to the physical I/O board. If you examine the value of the variable at other points in the POU execution, you may see a different value for the variable, because the force has not yet been applied at that point. Also, if you are seeing unexpected values for the variable, you may want to use the I/O Configurator to associate the I/O board containing points you are going to force with a particular task (see **“Related Task”** field in the I/O Configuration Wizard); this will ensure that execution of the task is coordinated with data updates to the I/O point.

Temporarily Overwriting a Variable's Value

To **overwrite** a variable, means that the user changes the variable's value, however, there is nothing to prevent logic in the control strategy from subsequently changing it.

- In Debug Mode, go into the worksheet for one of your POUs, for example, one of the graphical worksheets, and find the variable you want to overwrite.
 - Double-click on the variable that you want to overwrite. The Debug: RTU Resource dialog box will appear. (See page 110).
 - Specify the new value you want to send to the variable. (For REAL variables, enter a value in the **“Value”** field; for BOOL variables, select either **“TRUE”** or **“FALSE”**. After specifying the value, click on the **[Overwrite]** button, and the new value will overwrite the current value of the variable.
-

Note:

This only temporarily changes the variable's value. Logic in your ControlWave project can change it to a value different from the value you specified.

Setting a Breakpoint

A **breakpoint** is used to temporarily halt execution of the ControlWave at a particular spot in the executing project. Once a breakpoint is activated, the user can manually 'step through' the code, to observe the values of variables at each step of the program execution.

- In Debug Mode, go into the worksheet for one of your POUs, for example, one of the graphical worksheets, and double-click on a particular point in the code, where you want to set a breakpoint, for example, on a variable as it enters a function block. The Debug: RTU Resource dialog box will appear (see page 110).
- In the 'Breakpoint' section of the dialog box, click on the **[Set BP]** button to set the breakpoint. Execution of the project will halt at that point, and that area of the code will be highlighted in orange. If not already visible, call up the RTU_RESOURCE dialog



box by clicking on **“Online→ Project Control”** or by clicking on the Project Control icon.

- In the RTU_RESOURCE dialog box, you now can choose how you want to resume execution. There are three different choices:

Choice 1: Click on **[Go]** and the project will execute until it encounters another breakpoint, and then it will halt.

Choice 2: Click on **[Step]** and the project will execute the next item of code and halt, again, until you click on **[Step]** again. This allows the programmer to move through the execution step-by-step, and examine how variables are affected at each step of execution. The current location where execution has halted is highlighted in orange.

Choice 3: Click on **[Trace]** this is similar to **[Step]** except it shows even more detail by executing step-by-step through individual function blocks.



Clearing the Breakpoint(s)

When you are finished performing debugging operations, you must remove the breakpoints from your project, or it will be unable to execute properly.

In the Debug: RTU Resource dialog box, click on **[Reset BP]** to reset the current breakpoint (the one where execution is currently halted). To clear all breakpoints, click on **[Reset all BP]**.

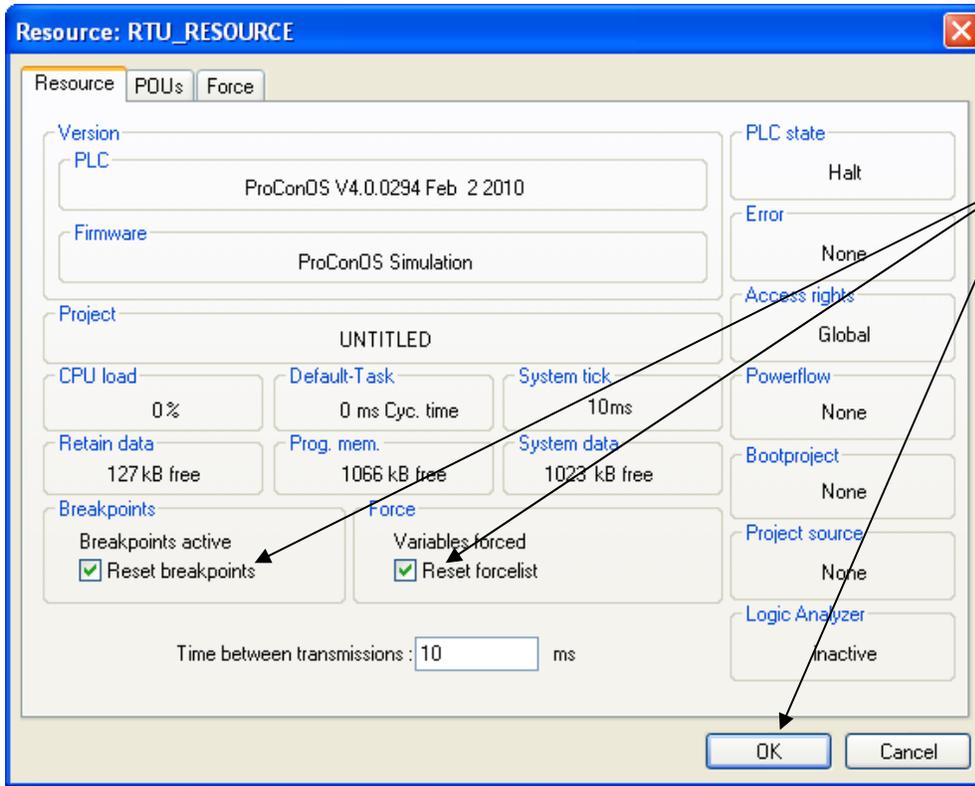
Verifying that Breakpoints and Forces Have Been Cleared

Before exiting Debug Mode, you should check to see that you have cleared all breakpoints, and force operations. Otherwise, these will remain active in your project, even after you exit Debug Mode.

To verify that these have been cleared, click on the **[Info]** button in the RTU_RESOURCE dialog box, and look at the 'Breakpoints' and 'Force' sections at the bottom of the Resource page. If it says 'No breakpoints active' and 'No variables forced' it is safe to exit Debug Mode.



If, however, you see 'Breakpoints active' or 'Variables forced', you must select the **“Reset breakpoints”** and **“Reset forcelist”** check boxes, and click on **[OK]**, prior to exiting Debug Mode, or the breakpoints and forces will remain set.



If you see “Breakpoints active” or “Variables forced” you should check the reset boxes and click “OK”.

Exiting Debug Mode

You must exit Debug Mode in order to make edits to your project. To exit on-line Debug Mode, click on the ‘Debug On/Off’ icon, shown above, or click on **Online→Debug**.

Downloading

Downloading is the process of transferring the compiled control strategy from your PC workstation, into the memory of the ControlWave controller.

Note:

Besides the control strategy, the compressed project source code (*.ZWT) can also be downloaded.

In ControlWave Designer, a user creates a control strategy to perform whatever system-specific job they want to use the controller for, and saves it as a ControlWave **project**. The project is compiled and the resulting computer code is then **downloaded** from the PC into the ControlWave's memory using either ControlWave Designer, or the OpenBSI Downloader.

When downloading the project directly from within ControlWave Designer, the user has a choice of downloading the project directly into dynamic memory for execution (SDRAM or SRAM depending on the platform), or downloading into the FLASH memory area (this is called downloading the **boot project**). When downloading from the OpenBSI 1131 Downloader, only the boot project may be downloaded.

A project can only *execute* from the dynamic memory area, but is lost in the event of a power failure or a program watchdog condition.

Because dynamic memory is not saved in these situations, a copy of the project is typically stored in the boot project area of FLASH memory. The boot project cannot execute from inside the FLASH memory area, but is automatically copied into the dynamic memory area during system re-boot (power restoration after power failure, restart following program watchdog condition.)

Typically, users download their project into the dynamic memory area only during system development and debugging. Once a control strategy is finalized, and has been tested fully, it should be downloaded as the boot project into FLASH memory.

Two Methods Available for Downloading

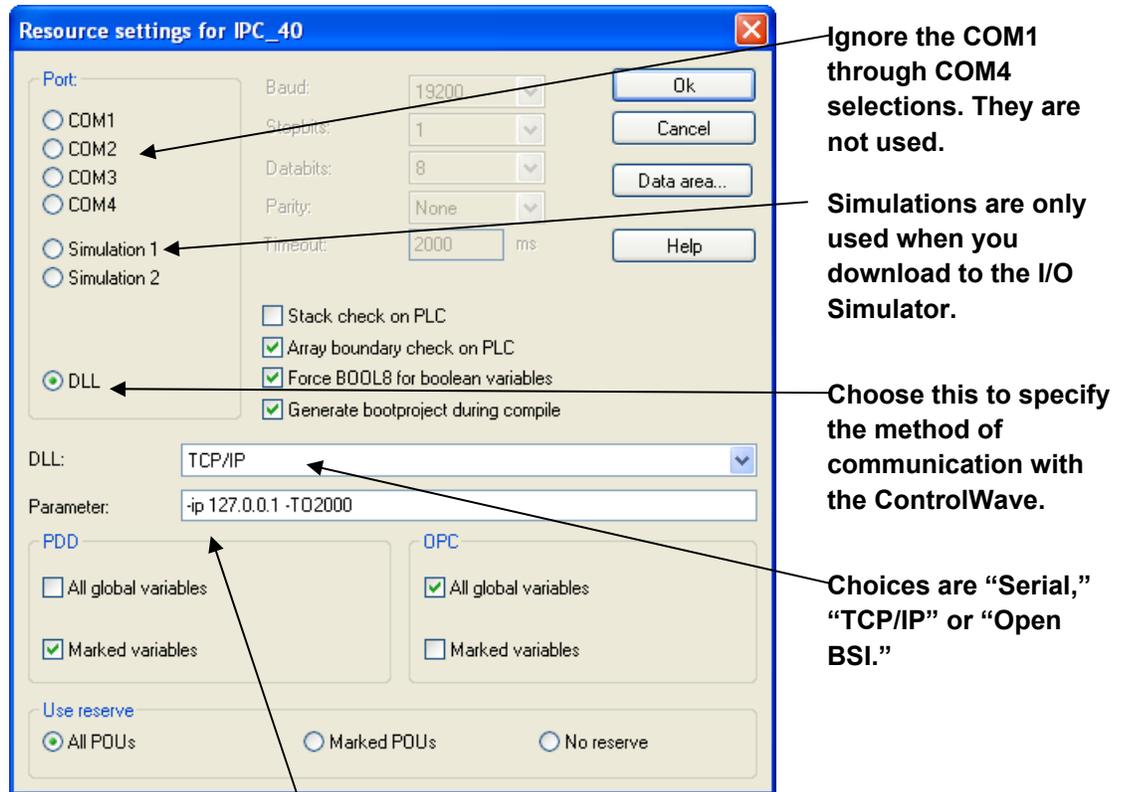
There are two ways to download your ControlWave project into the ControlWave series controller:

- Download the project from within the ControlWave Designer program.
- Download the project with the OpenBSI IEC61131 Downloader.

Downloading from within ControlWave Designer

Using the Resource Settings Dialog Box to Set up ControlWave Designer Communications

Step 1. Connect a cable between the PC workstation running ControlWave Designer, and the ControlWave. The type of cable used will vary depending upon which ports on the PC and on the ControlWave you have chosen. *NOTE: For ControlWave Designer TCP/IP or OpenBSI connections, the cable might not go directly to the ControlWave, it may go to a network, of which the ControlWave is part.*



Ignore the COM1 through COM4 selections. They are not used.

Simulations are only used when you download to the I/O Simulator.

Choose this to specify the method of communication with the ControlWave.

Choices are "Serial," "TCP/IP" or "Open BSI."

If you choose "Serial," for the DLL, specify the PC comm. port, baud rate, and timeout in milliseconds. If you choose "TCP/IP" specify the IP address of the ControlWave, and the timeout in milliseconds. If you choose "Open BSI" just specify the node name of the ControlWave.

Step 2. Open your ControlWave Designer project. For help on creating a project in ControlWave Designer, see *Getting Started in ControlWave Designer* (part number D301416X012).

Step 3. Call up the Resource Settings dialog box by *right-clicking* on RTU_RESOURCE: ControlWave in the project tree, and choosing "Settings" in the pop-up menu. *NOTE: This dialog box is NOT accessible when on-line communication is already in progress.*

- Step 4. In the 'Port' box, choose **"DLL"** and the **"DLL"** list box and **"Parameter"** field will be activated. NOTE: **"DLL"** will be grayed out and unavailable if you do NOT have a software copy protection key (dongle) plugged into the parallel port of your PC.
- Step 5. In the **"DLL"** list box, choose one of the three available DLLs ('Serial', 'TCP/IP', or 'OpenBSI'), and enter the **"Parameters"** appropriate to that DLL.

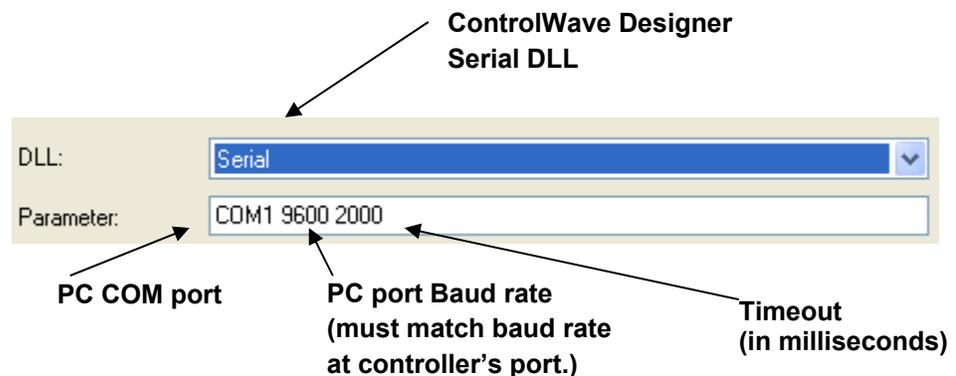
'Serial'

If you choose 'Serial', this means you will use the ControlWave Designer Protocol (Serial DLL) to communicate with the ControlWave.

In order to use this DLL, the ControlWave key switch must be in the 'LOCAL' position.

The cable connection must be to one of the ControlWave's serial COM ports.

In the **"Parameter"** field (shown below) specify the PC COM port (not the ControlWave COM port), the baud rate at which communications will occur (which must match the baud rate configured at the ControlWave) and a timeout for the port in milliseconds.



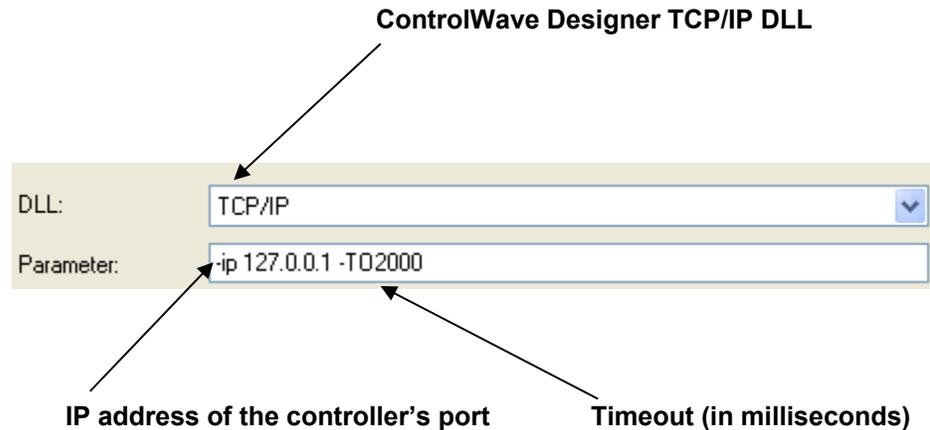
'TCP/IP'

If you choose 'TCP/IP', this means you will use the ControlWave Designer Protocol (TCP/IP DLL) to communicate with the ControlWave.

In order to use this DLL, the ControlWave key switch must be in either the 'LOCAL' or 'REMOTE' position.

The ControlWave must be reachable via TCP/IP, from this PC.

In the "Parameter" field (shown below) specify the IP address of the ControlWave port to which the connection has been made, and a timeout for the port in milliseconds.

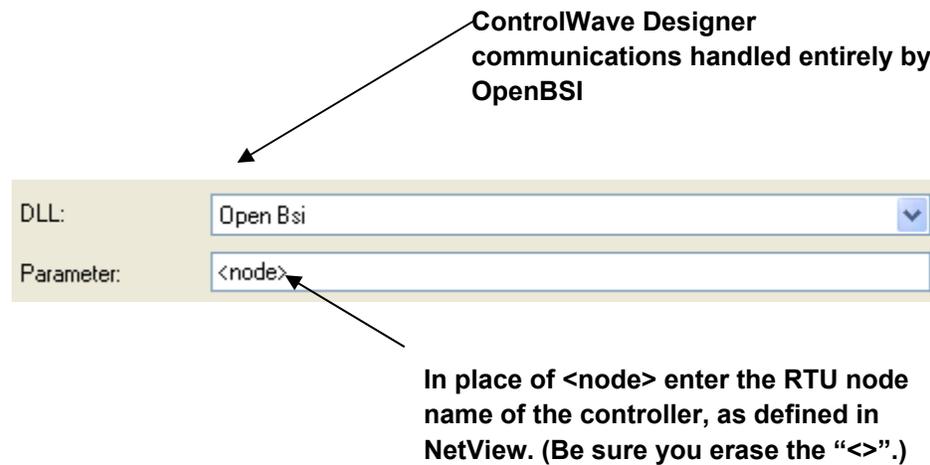


'OpenBSI'

If you choose 'OpenBSI', this means OpenBSI will handle all communications between ControlWave Designer, and the ControlWave. For this to work, you must have already configured OpenBSI, and included the ControlWave controller in an OpenBSI network.

The ControlWave key switch must be in either the 'LOCAL' or 'REMOTE' position.

In the "Parameter" field (shown below) replace the '<node>' with the RTU node name (as defined in the OpenBSI network) for the ControlWave.



Step 6. Click on **[Ok]** to exit the Resource Settings dialog box. You can now proceed to download your ControlWave project, or you can enter debug mode.

Downloading Your ControlWave Project from Within ControlWave Designer

Before downloading a project into the ControlWave, you must have compiled all of your edits and specified the communications method using the Resource Settings dialog box (see above).

If the ControlWave is connected to an actual running process, we recommend you download and test the project in the I/O Simulator *first*.

⚠ DANGER

Users should never attempt to download an *untested* program into a controller if the controller is currently connected to a running plant or industrial process. Safeguards must be taken prior to downloading to ensure that the controller is isolated from the process and I/O is disconnected. Failure to take such precautions could result in injury to persons or damage to property.

Step 1. Click on **Online** → **Project Control**

Step 2. The RTU_RESOURCE dialog box will appear. If there is already a project running in the ControlWave, you can optionally stop it first, before proceeding with the download, by clicking on the [Stop] button. Provide your username and password, if prompted. Now click on the [Download] button to call up the Download dialog box.



If you already have a project running in the controller, click here to stop it, *first*.

Click here to call up the Download dialog box.

Step 3. Click on the **[Download]** button on the left side 'Project' portion of the dialog box, and the compiled project code will be downloaded in the SDRAM memory (or SRAM) of the ControlWave. (There are several other options for downloading; see the figure on the next page).

Note

For many ControlWave platforms (GFC, XFC, Express) there is no SDRAM. The control strategy file executes in Static RAM (SRAM).

To download the bootproject (*.pro) at the same time as you download the ControlWave project, select this.

Click here to download your compiled project. This is the actual executable code which runs in the ControlWave. The project code executes in either SDRAM or SRAM.

Click here to download just the compiled project code into the bootproject area of FLASH memory. This is preserved in the event of a power failure.

To download the zipped ControlWave source file (*.zwt) at the same time as you download the ControlWave project, select this.

You should check "Include OPC data" if you want to use ObjectServer or you want to use Signal Extractor to create a database.

Click here to download just the zipped ControlWave source file (*.zwt). This is the file and libraries you actually edit with ControlWave Designer, NOT the compiled code which executes in the ControlWave.

User libraries are user-created collections of functions and function blocks.

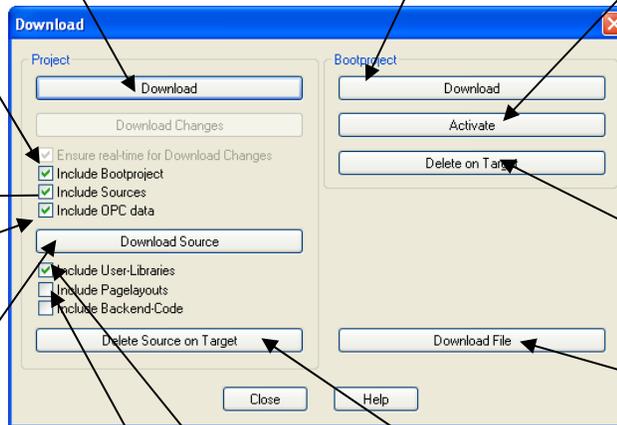
Pagelayouts are the graphical elements of your project e.g. the project tree.

"Activate" copies the bootproject into SDRAM or SRAM (depending on the platform). This happens automatically on power-up.

This deletes the bootproject.

This can be used to download other files (e.g. web pages) to the FLASH area of the ControlWave.

Click here to delete any existing *.ZWT file already in the ControlWave.



Step 4. In the RTU_RESOURCE dialog box, start the project execution by clicking on either the [Warm] or [Cold] buttons.

The [Warm] button performs an application warm start. The [Cold] button performs an application cold start. An **application warm start** means that the project in SDRAM is started from the beginning of its cycle, using saved values for variables marked "RETAIN" from the Static RAM (SRAM). Application warm starts are performed whenever there is no version mismatch between the project, and the retain values, and there was no *system* cold start. If a *system* cold start occurred, however (i.e. loss of battery power to the SRAM, or static memory SRAM Control Switch set OFF) all data in static RAM is gone, so an **application cold start**

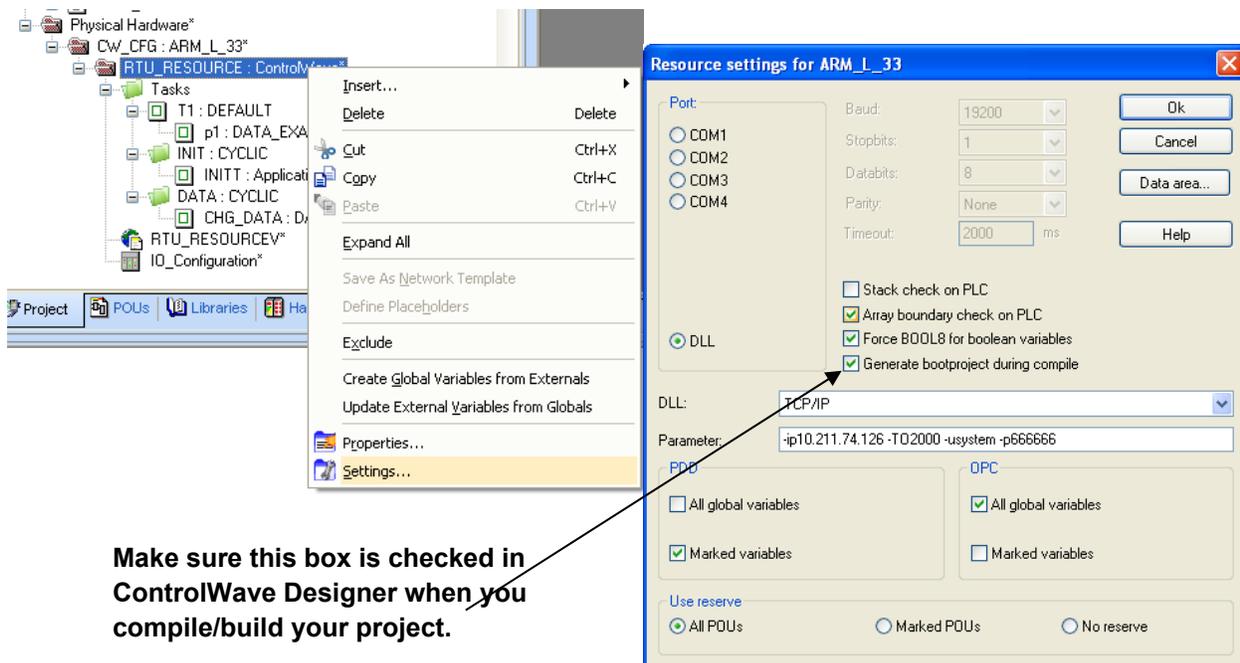
must be performed. In this case, the project in SDRAM is started from the beginning, and all variables are set to their initial values. Application warm starts and cold starts can also be performed by the user after downloading a project from within ControlWave Designer by choosing the **[Cold]** or **[Warm]** buttons in the RTU Resource dialog box. For more information on this subject, see 'Memory Usage' later in this manual.

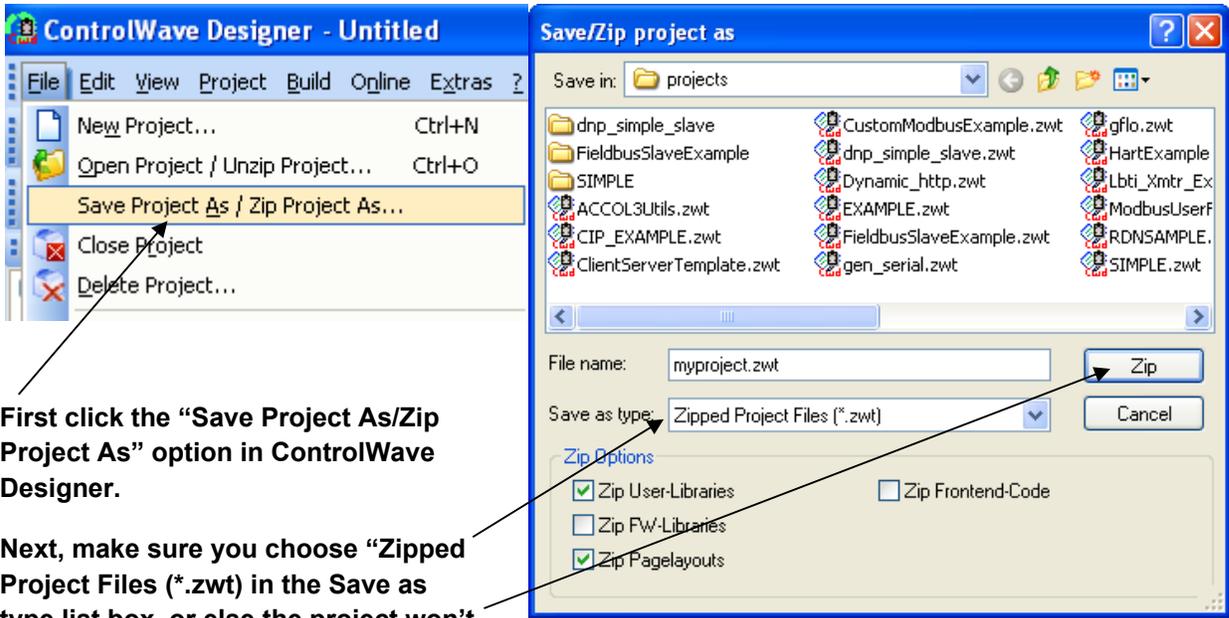
Downloading using the OpenBSI ControlWave Downloader

Before You Begin

There are certain things you must do before you can download to a ControlWave-series controller.

- You must save your project as a ControlWave project *.MWT file.
- You must generate a boot project file during compilation in ControlWave Designer. To do this, you must check the **Generate bootproject during compile** box for your resource.
- You must generate a zipped project file (*.ZWT) in ControlWave Designer. One way you can do this is to manually save your ControlWave project as a zip file:

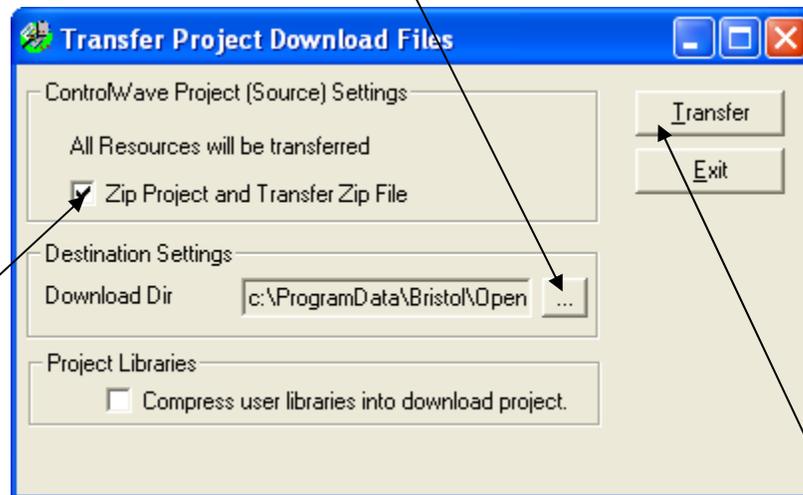




First click the “Save Project As/Zip Project As” option in ControlWave Designer.

Next, make sure you choose “Zipped Project Files (*.zwt) in the Save as type list box, or else the project won't be zipped.

Use the “...” button to specify the directory which will hold your download files. When you initiate a transfer, the utility creates a sub-directory of the download directory to hold the boot and zip files for this particular project.



If you didn't generate a ZWT file yet, check this box and the utility does it for you.

Click here to start the transfer.

You must transfer the bootfile and zip file for this project to a *sub-directory* of whichever directory you want to use for downloads. You can accomplish this if you click **Build** → **Transfer Download Files** in ControlWave Designer. In this utility, you must specify the download directory in the **Download dir** field.

If you check **Zip Project and Transfer Zip File** (default), the system zips the current project automatically, in preparation for the transfer. If you select the **Compress user libraries into download project** option, the system zips the user libraries and includes them in the zip project.

Note:

Zip Project and Transfer Zip overwrites any pre-existing zip file for this project. To prevent this, you can disable the option, however, if you do, you must have a previously created zip available for transfer.

When you finish making selections, click **Transfer** and the file transfer begins.

If your ControlWave-series node includes a key operated RUN / REMOTE/ LOCAL switch, you must turn the switch to either the REMOTE or LOCAL position, depending upon how the PC connects to the ControlWave. Downloading CANNOT occur with the switch in the RUN position.

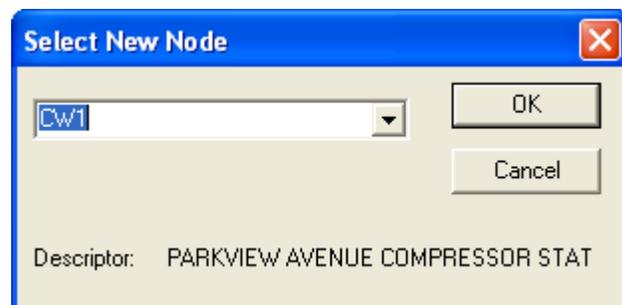
Starting the ControlWave Downloader

There are two methods for starting the ControlWave Downloader:

Method 1:

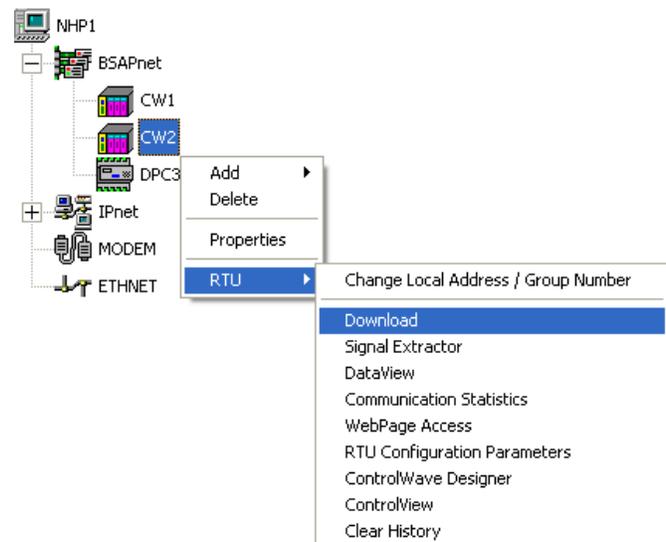
Click **Start** → **Programs** → **OpenBSI Tools** → **ControlWave Tools** → **ControlWave Downloader**.

The Select New Node dialog box opens. Use the list box to select the node which you want to download to; then click **OK**, and the Downloader opens.



Method 2:

The second method is to *right-click* on the icon for the controller you want to download, in the NetView tree, and choose **RTU** → **Download** from the pop-up menu.



Using the ControlWave Downloader

When the ControlWave Downloader dialog box opens, complete the fields as described, below:

Enter the proper username and password for this controller.

Click here to start the download.

RTU node name (as it appears in the NetView tree)

Check this to allow the ZWT file to download.

Shows the progress of the download.

Use this browse button to choose the sub-directory containing your bootfile.pro and

Use this browse button to choose the sub-directory containing user files. You can use the ControlView utility to retrieve these.

Use this browse button to choose the sub-directory containing user files. You can use the ControlView utility to retrieve these.

Check this box to download user files (.HTML, etc.) which the ControlView utility can retrieve later.

When downloading a project, click “Warm Boot” to perform a warm download (project is started from the beginning using values saved as RETAIN – if project hasn’t changed to a degree that those values don’t apply). If you de-select “Warm Boot” a cold download occurs (project is started from the beginning

When the fields are completed, click **Begin** to start the download. The fields/buttons in this dialog box are:

Field	Description
Node	This displays the node name (as it appears in the NetView tree) for this ControlWave-series controller.
Username, Password	Enter a valid username/password combination for this ControlWave-series controller.
Project Path	Enter the path of the project that the Downloader will download to this controller, or use the Browse Bootfile button to locate it. (The path must be a sub-directory of whichever directory you specified for downloads (Download Dir) in the Transfer Download Files dialog box) The project files consist of the .PRO boot file, generated when you compile your ControlWave project, and the zip file (*.zwt) containing the project source. Note: If your

Field	Description
	project includes multiple resources, each one has a different path, and you must choose the appropriate one.
User Files Path	Enter the path of the folder containing files you want to download to the user files area of the ControlWave, or use the Browse Path button to locate it. (See Download User Files, below).
Begin	Click here to start the download.
Cancel	Click here to exit down the 1131 Downloader.
Warm Boot	When you don't select this check box, all variables initialize as part of the download, and the project restarts. When you choose Warm Boot, any variables configured as RETAIN do not re-initialize as part of the download, however, all other variables initialize, and the project restarts from the beginning of its cycle.
ZipFile	When you select this option, the download operation includes the zipped project file (*.ZWT).
Download User Files	The ControlWave can store user files (*.ZIP, *.HTML, etc.) in flash memory, for later retrieval using the ControlView utility. You must place the user files you want to download to the ControlWave in the folder identified by the User Files Path field. Note: This feature was added in OpenBSI 5.3 Service Pack 2.

Creating Download Scripts for Batch Downloading of ControlWave Controllers

Optionally, you can create download scripts which allow you to download files to ControlWave controllers using a single command.

You create download scripts as ASCII text files, with the file extension of *.RDL, and store them in the **Downloads** sub-directory of your OpenBSI directory.

Each line of the download script, defines the downloading parameters for a single ControlWave controller. The syntax of a line of the download script is:

```
nodename,filetype,startup,includezip,source_path
```

where:

nodename is the name of the ControlWave controller you want to download. This name must match the name you define in NetView. (This is the only required field.)

filetype specifies the kind of file you want to download. filetype must be either:

- P Download a ControlWave project (default)
- F Download a user file (used with ControlView)

<i>startup</i>	specifies whether the system should perform a warm boot upon completion of the download. <i>startup</i> must be either:
	Y Perform a warm boot (default)
	N Do not perform a warm boot
<i>includezip</i>	specifies whether or not the Downloader should also download the zipped ControlWave project (*.ZWT). <i>includezip</i> must be either:
	Y Include *.ZWT with the download
	N Do not include *.ZWT with the download (default).
<i>path</i>	specifies the source folder containing the file you want to download. If you download a project, this must be the directory containing <i>bootfile.pro</i> . If you download user files for use with ControlView, this must be the folder containing those files. If the folder name contains spaces, you must surround it with quotation marks “ ”. If you enter nothing here, the Downloader uses OpenBSI Application Parameter defaults.

Example RDL File:

```
RPC1,P,Y,Y,C:\ProgramData\Bristol\OpenBSI\My downloads"
```

```
RPC2,P,Y,Y,C:\ProgramData\Bristol\OpenBSI\My downloads"
```

```
RPC3,P,Y,Y,C:\ProgramData\Bristol\OpenBSI\My downloads"
```

```
RPC4,P,Y,Y,C:\ProgramData\Bristol\OpenBSI\My downloads"
```

Starting the Download Script

To start the download script you create, click on File Open Script within the ControlWave Downloader, then choose the RDL file that contains the download script.

You can also run download scripts from the command line prompt according to the following syntax:

```
dl1131 script_name username password
```

where:

script_name is the name of the RDL file (omitting the RDL extension)

username password is a valid username/password combination for the first RTU in the script. The named user must have privileges sufficient to download.

For example, to run the download script *myloads.RDL* where the first RTU in the RDL file has a username/password combination of THOMAS BOB276, type the following:

```
dl1131 myloads THOMAS BOB276
```

Running the ControlWave Downloader from the Command Line:

Optionally, you can start ControlWave Downloader from the DOS command prompt. Follow the syntax rules below; optional switches appear in brackets “[].” The command for this is:

```
dl1131 node [file] username password
```

where:

- | | |
|------------------------------|---|
| <i>node</i> | is the RTU node name as defined in the NETDEF files. If no file is specified, the Downloader uses the file specified in the RTU Properties in NetView. |
| <i>File</i> | is the basename of the ControlWave project. You can omit the .PRO or .MWT extension. When you specify a file, you override any filename specified in the RTU Properties in NetView. If the filename includes spaces, you must surround it with quotation marks “ ”. |
| <i>username
password</i> | is a valid username/password combination for this RTU. The user you specify must have sufficient privileges to perform the download. |

Expanded BSAP (EBSAP) Communications

A Bristol Synchronous / Asynchronous Protocol (BSAP) network enforces an absolute limit on the number of nodes (remote process controllers) which may be addressed on the level below a given node or OpenBSI Workstation. That limit, from all master ports combined, is **127** nodes. Certain applications (particularly those involving large numbers of radio remotes operating on the same frequency) may require larger numbers of nodes to be addressed through the same port. To address more than 127 nodes through a given master yet still not violate the BSAP limit, you must use a technique called *expanded BSAP* (or EBSAP).

Expanded BSAP allows the creation of **virtual nodes** below an **EBSAP Master Port**, and each of these virtual nodes can address up to 127 actual nodes. The virtual node, itself, exists only as a software structure in the **EBSAP Master** node above it; there is no additional physical remote process controller involved. The presence of the virtual node, however, introduces an intermediate level into the network which allows up to 127 actual physical nodes to be addressed below it, while only one node, the virtual one, is counted on the intermediate level. Since a real, physical remote process controller on level n of the network can address 127 nodes below it (all on level $n+1$), if each of these nodes were virtual nodes, and each virtual node had 127 real, physical nodes below it (on level $n+2$) expanded addressing would theoretically allow 127 groups of slave nodes, with each group containing up to 127 nodes, all addressed by a single master node. In fact, they could all be on the same master port. That's 127 X 127 or a total of 16,129 nodes!

Important

Although the expanded addressing scheme *theoretically* allows 16,129 nodes, other *practical* limitations (such as running out of memory in the master node, the ability of OpenBSI to support only up to 4,999 nodes, limitations on the number of data collection templates in the HMI, and the unacceptable length of time required to poll so many nodes) would rule out such a large number of nodes.

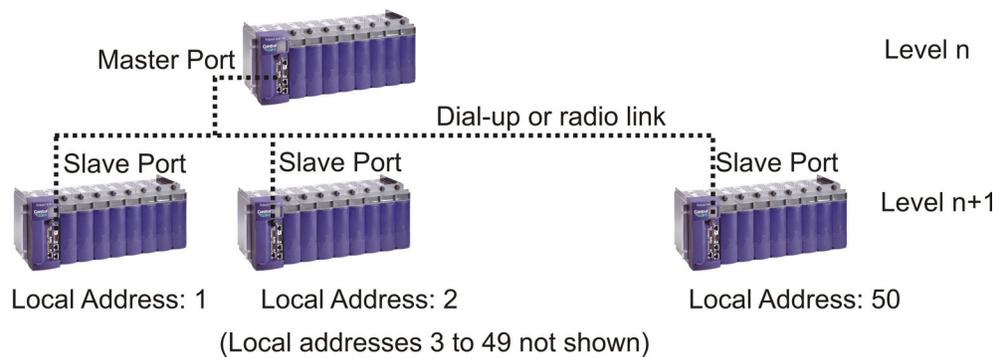
On some systems with radio remotes, where very long polling rates (several minutes, at least) are acceptable, it may be practical to use expanded BSAP. In other time-critical applications, even fifty nodes (far fewer than 127, and thereby not requiring expanded BSAP) may be too many to support fast data updates. It all depends on the capabilities of your network.

CAUTION

The size and composition of a network and the decision to use expanded BSAP must be made **only** after you have carefully examined of your system requirements and have **fully considered** how expanded BSAP may impact network performance.

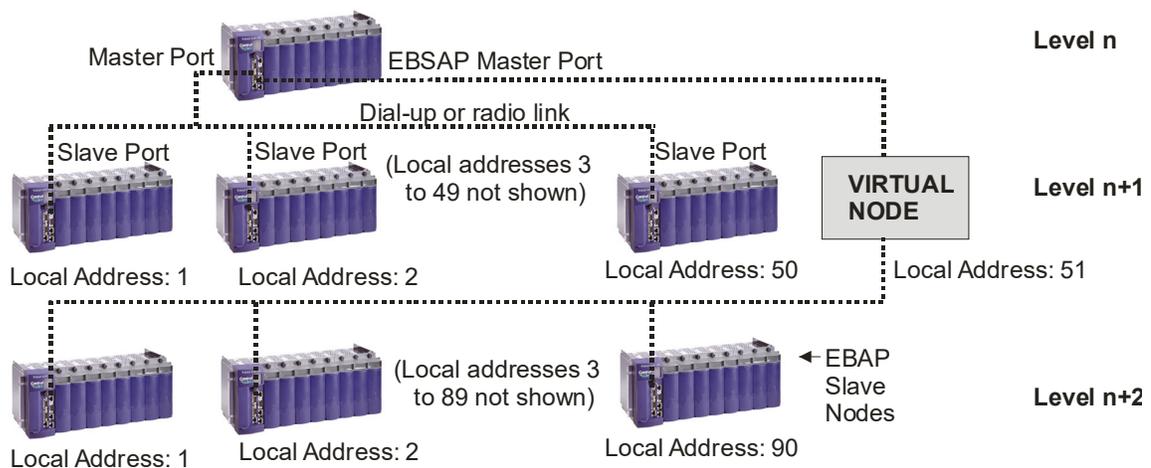
Expanded BSAP – The Concept

The following figure shows a portion of a BSAP network which does **not** use EBSAP. It has a single master node (on level n) with 50 slave nodes below it (on level $n+1$). These slave nodes use local addresses 1 through 50 (though there isn't room to show all of them in this picture.)



Because of new system requirements, it is decided that 90 additional slave nodes must be addressed through this master node, for a total of 140 slave nodes. This presents a problem because with fifty slave nodes already present, only 77 additional nodes could normally be added ($50 + 77 = 127$).

To get around this limitation, expanded BSAP will be used (see the figure below). An **EBSAP Master Port** will be added that will allow communication with virtual nodes. A **virtual node**, with a local address of 51, will be added to the network (on level $n+1$). The virtual node can address the 90 new real, physical nodes on the level below it (level $n+2$). These are called Expanded Addressing Slave (EASlave) nodes (or just **EBSAP slave nodes**).



Since the virtual node is really a phantom controller, existing only in software, the EBSAP scheme has allowed a single EBSAP Master node at level n , to address 140 nodes; 13 more than would normally be allowed.

General Requirements for Expanded BSAP (EBSAP):

Every EBSAP network must have:

- An EBSAP Master node.

An EBSAP Master node can be any of the following:

1. An OpenBSI Workstation that has been configured with an **EBSAP communication line**. Virtual nodes must be on level 1 of the network, and EBSAP slaves must be on level 2 of the network.
2. Any ControlWave-series controller with version 04.50 or newer firmware that has been configured with an **EBSAP Master Port**. Control and status arrays must also be defined to handle polling for EBSAP slaves.
3. Certain Network 3000-series controllers (DPC 3330, DPC 3335, RTU 3310, or RTU 3305 with AG, RMS00, PLS00, LS500 or newer firmware). NOTE: This manual does NOT cover how Network 3000-series controllers are used in EBSAP. For information on this subject, please refer to the *Expanded Node Addressing* section of the *ACCOL II Reference Manual* (document# D4044).

- One or more **virtual nodes**

All the nodes *immediately below* an EBSAP Master port or on an EBSAP communication line must be virtual nodes. Virtual nodes are just software structures that reside in the EBSAP Master. They are defined in NetView, similar to any other RTU, but they are not actual hardware.

- From 1 to 127 **EBSAP slave nodes** underneath each virtual node

An EBSAP slave node can be any ControlWave-series controller or any ACCOL II-based Network 3000-series controller.

The EBSAP slave node must be assigned to the proper EBSAP group. EBSAP slave nodes underneath the *first* virtual node on an EBSAP Master port must be assigned to Group 0; EBSAP slave nodes underneath the *second* virtual node on an EBSAP Master port must be assigned to Group 1; and so on.

The following rules must be adhered to when using EBSAP:

- All nodes, whether master, slave, or virtual, must be defined in the current NETDEF file.
- All slaves on an EBSAP Master Port, which are on the level immediately below the node containing the port, must be virtual nodes.
- All slaves of the virtual nodes must be real remote process controllers; i.e. a virtual node cannot have a virtual slave.

- Just like local address assignments, the choice of group numbers is important, and should NOT be done randomly. The **EBSAP Group number** identifies which virtual node on a given EBSAP communication line is above a particular RTU. See *Specifying the Group Number for an RTU* later in this section, for details.
- Peer-to-peer communication of signal lists and data arrays is only supported between nodes within the same group, and immediately below the same virtual node.
- Poll periods and timeouts must be set carefully based on the size of your EBSAP network.

Creating an EBSAP Master

As noted earlier, an OpenBSI Workstation or a ControlWave controller *can* serve as an EBSAP Master. Certain Network 3000 controllers can also serve as EBSAP Masters; they are not discussed in this manual.

Note:

For information on using Network 3000 controllers in an EBSAP network, refer to *Expanded Node Addressing* section of the *ACCOL II Reference Manual* (document# D4044).

OpenBSI Workstation is EBSAP Master

To make an OpenBSI Workstation the EBSAP Master, you must define an EBSAP communication line that covers the address range of the virtual nodes.

Defining an EBSAP communication line:

There are two methods for defining an EBSAP communication line:

Method 1:

In NetView, drag an EBSAP line icon from the Toolbox into your network hierarchy. This will activate the Comm Line Wizard, from which you can proceed to define the EBSAP line as you would a regular BSAP line.



Method 2:

If you're already in the Comm Line Wizard, choose 'EBSAP Line' as the node type and then continue to define it as you would a regular BSAP line.

Select EBSAP Line



ControlWave-series Controller is the EBSAP Master

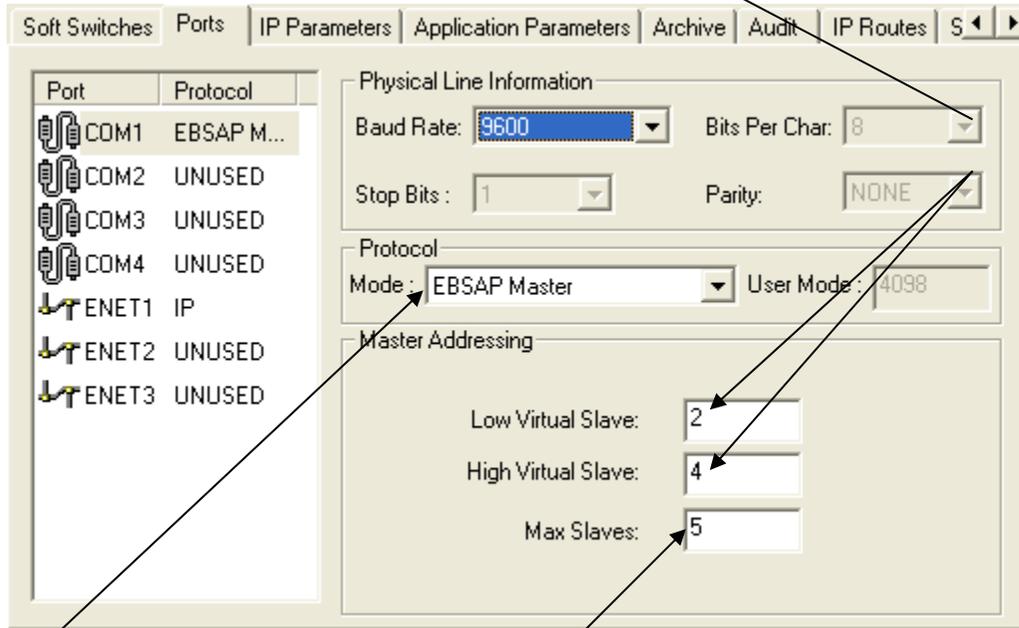
Any ControlWave-series controller with 04.50 or *newer* firmware can serve as an EBSAP Master. To be an EBSAP Master, the ControlWave-series controller must be configured with an EBSAP Master Port, as well as control and (optionally) status arrays.

Defining an EBSAP Master Port

An EBSAP Master Port is configured from within the Flash Configuration Utility.

- Choose 'EBSAP Master' as the **"Protocol Mode"**.
- Specify the baud rate for the port in the **"Baud Rate"** field.
- Using the **"Low Virtual Slave"** and **"High Virtual Slave"** fields, specify the range of local addresses used by the virtual nodes. For example, if the virtual slaves use local addresses 5 through 9 on this EBSAP Master Port, then enter '5' for the **"Low Virtual Slave"** and '9' for the **"High Virtual Slave"**.
- Set **"Max Slaves"** to the maximum number of *real* slave nodes that will be on the level immediately below any one of the virtual nodes on this port. For example, if you have 3 virtual nodes on this port, and one will have 20 nodes underneath it, another will have 15 nodes underneath it, and the third will have 27 nodes underneath it, then **"Max Slaves"** would be set to 27, since that is the maximum under any one virtual node. NOTE: The value of **"Max Slaves"** must be consistent with the maximum size of your network, as defined in NetView; i.e. if you mustn't specify more slaves than are supported by the network levels you defined previously. In addition, the **"Max Slaves"** value must be used later when you define the size of the control and status arrays for your port.
- When finished, save the port definition to the RTU by clicking on **[Save to Rtu]** and reset the controller for the change to take effect.

Defines the local address range of the virtual nodes on this port.



Choose "EBSAP Master"

Maximum number of real nodes allowed underneath one of these virtual nodes.

Configuring the Control and Status Arrays

There are four different arrays that are useful in EBSAP. Basically, they are used to turn ON/OFF polling for nodes, and to report whether communications are working for particular nodes.

Note:

ll these arrays must be registered using REG_ARRAY function blocks, and must be marked as PDD variables so they can be accessed by external programs (e.g. DataView).

_SLAVE_DEAD array and _SLAVE_POLL_DIS array

The _SLAVE_DEAD array provides an indication whether any responses have been received for slaves of the EBSAP Master. Because all the slaves immediately below the EBSAP Master are virtual nodes, the indication of responses received will actually deal with the EBSAP slaves under each virtual node. If there are any EBSAP slaves responding through a

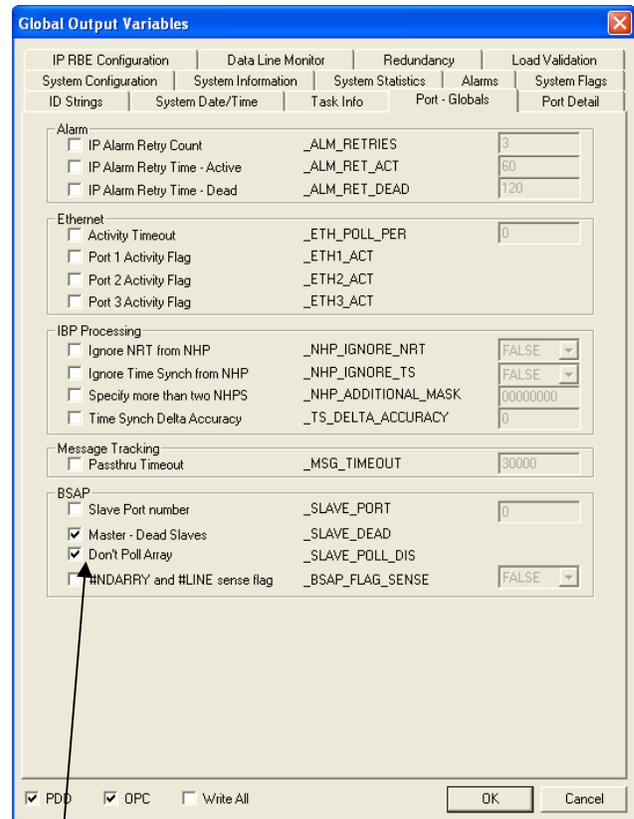
particular virtual node, the virtual node is considered 'alive'; otherwise, it is considered 'dead'.

The `_SLAVE_POLL_DIS` array allows the programmer to selectively turn ON/OFF polling for the slaves of the EBSAP Master. Because all the slaves immediately below the EBSAP Master are virtual nodes, turning OFF polling for a particular virtual node will actually turn OFF polling for all the EBSAP slaves under that virtual node..

Because both the `_SLAVE_DEAD` array and `_SLAVE_POLL_DIS` array are also used in *standard* BSAP systems, they are automatically present in your project. You need to generate the system variables for them, however, in order to make reference to them.

To configure the `_SLAVE_DEAD` and `_SLAVE_POLL_DIS` arrays do the following:

1. On the 'Port Globals' page of the System Variable Wizard, check the **"Master - Dead_Slaves"** (`_SLAVE_DEAD`) and **"Don't Poll Array"** (`_SLAVE_POLL_DIS`) boxes. The arrays should also be marked as **"PDD"** to allow for collection by OpenBSI.



Check the boxes for **"_SLAVE_DEAD"** and **"_SLAVE_POLL_DIS"**

2. This will cause a new worksheet to be created called SYS_VAR_WZ_TYPES. This worksheet contains the data types used by the _SLAVE_DEAD and _SLAVE_POLL_DIS arrays. In order to ensure that the new data type is fully accessible, please re-build your project using the command Build → Rebuild Project.

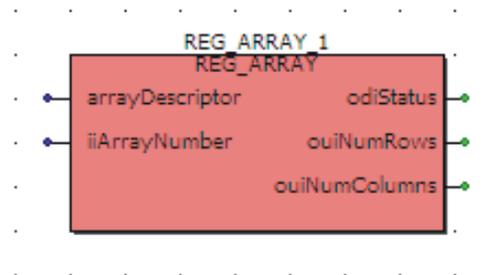
```

1 | (*
2 | BBI_SYS_VAR_WIZ_TYPES_START
3 | NOTE: The System Variable Wizard will maintain these
4 | Please do not attempt to modify this section :
5 | *)
6 | TYPE
7 |   BA_10 : ARRAY [0..9] OF BYTE;
8 | END_TYPE
9 | TYPE
10 |   B_1_1 : ARRAY [1..1] OF BOOL;
11 | END_TYPE
12 | TYPE
13 |   B_127 : ARRAY [1..127] OF B_1_1;
14 | END_TYPE
15 | (* BBI_SYS_VAR_WIZ_TYPES_END *)
16 |

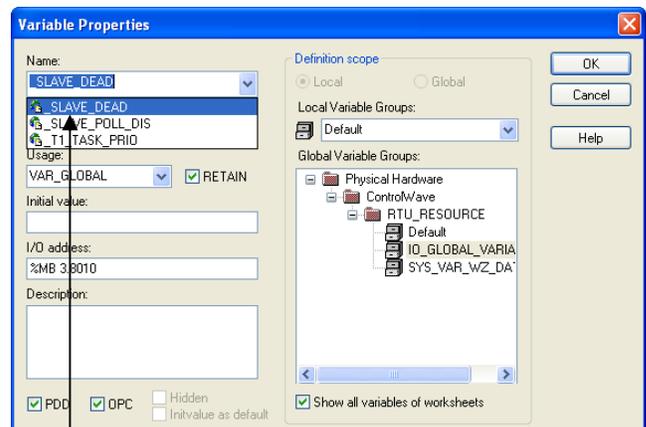
```

_SLAVE_DEAD and _SLAVE_POLL_DIS use the data type 'B_127'

3. For each of these two arrays (_SLAVE_DEAD and _SLAVE_POLL_DIS) you will want to register the arrays, so they can be made accessible to OpenBSI programs such as DataView or Harvester. To do this, use the Edit Wizard in ControlWave Designer to insert a REG_ARRAY function block (1 for each array).



- a. The **arrayDescriptor** parameter should be set to the name of the array (either _SLAVE_DEAD or _SLAVE_POLL_DIS in this case.)
- b. The **iiArrayNumber** parameter assigns an array number to the array, so it may be requested by external programs such as OpenBSI's DataView or Harvester.
- c. The remaining



Choose either “_SLAVE_DEAD” or “_SLAVE_POLL_DIS” depending upon which one you are registering.

parameters,
odiStatus,
ouiNumRows,
ouiNumColumns
need only be
assigned variable
names; since they are
output parameters.

When you have completed this configuration for both arrays, and compiled and downloaded the project, the `_SLAVE_DEAD` and `_SLAVE_POLL_DIS` arrays will be operational.

Note:

The interpretation of the BOOL array elements in all the control and status arrays can be toggled based on the value of the `_BSAP_FLAG_SENSE` system variable. By default, `_BSAP_FLAG_SENSE` is FALSE. By default then, in the `_SLAVE_DEAD` array, a TRUE BOOL value means that the associated virtual node is dead; i.e. there are no 'live' EBSAP slave nodes for that virtual node. Also, by default, a TRUE BOOL value in `_SLAVE_POLL_DIS` turns polling OFF for a slave node.

If, however, the user changes the value of `_BSAP_FLAG_SENSE` to TRUE, the logic for interpreting the values of the array elements is reversed; a TRUE BOOL value in the `_SLAVE_DEAD` array would mean that there is at least one 'live' EBSAP slave node for this virtual node; similarly a TRUE BOOL value in `_SLAVE_POLL_DIS` would turn polling ON for a slave node.

`_Px_DEAD_ARRAY` and `_Px_DISABLE_ARRAY`

The `_Px_DEAD_ARRAY` and `_Px_DISABLE_ARRAY` system variables define the registered number of user defined arrays that handle reporting on whether polling is successful to EBSAP slaves on this port, and whether polling for the EBSAP slaves on this port should be enabled or disabled.

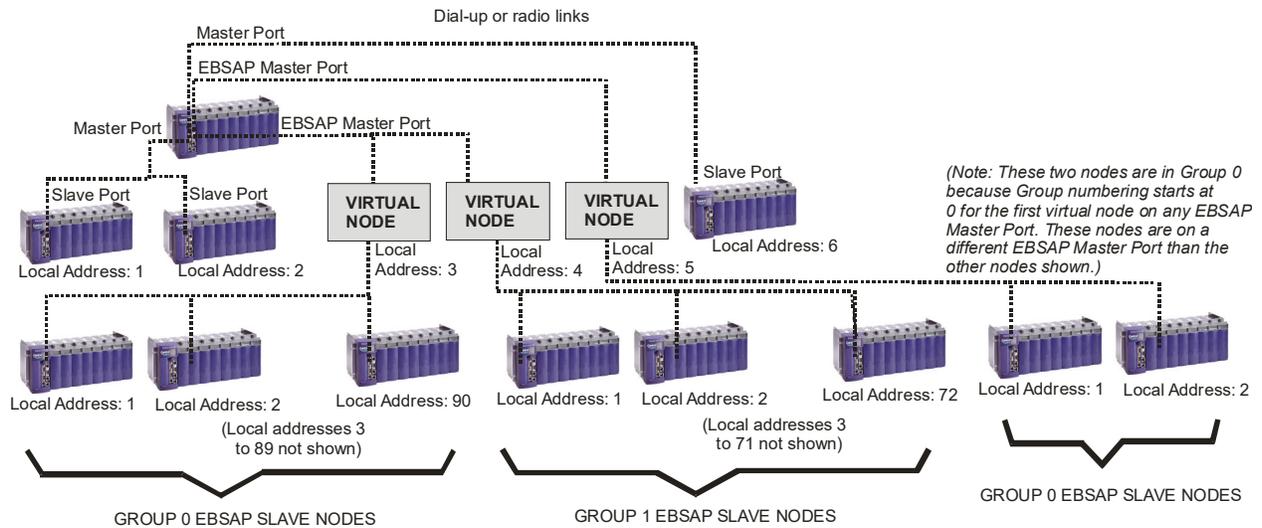
The user-defined array referenced by `_Px_DEAD_ARRAY` performs a similar function to the `_SLAVE_DEAD` array, except that instead of reporting on the virtual nodes, it reports on whether responses have been received from any EBSAP slave on this particular EBSAP Master Port. Users must define the data type for this array, based on the maximum number of EBSAP slave nodes for any virtual node on this port, and by the local address range of the virtual nodes used on this port.

Let's look at the network, below. It has one EBSAP Master port (Port 2) with 2 virtual nodes, and another EBSAP Master port (Port 3) with 1 virtual node. For Port 2, one virtual node has 90 EBSAP slave nodes; and the other has 72 EBSAP slave nodes. The maximum number of EBSAP slave nodes under any virtual node on Port 2 is therefore 90 so "Max Slaves" for Port 2 is set to 90. The single virtual node under Port 3 has just 2 EBSAP slave nodes, so the

maximum number of EBSAP slave nodes for Port 3 is 2, and “Max Slaves” for Port 3 is set to 2.

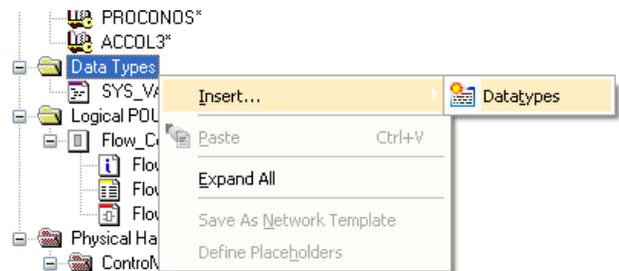
Important:

Because these arrays use dimensions of ‘Max Slaves’ as defined in the Flash Configuration Utility; the appropriate dimension of the array must match that value exactly. Sizing the array larger or smaller, will prevent EBSAP from working.



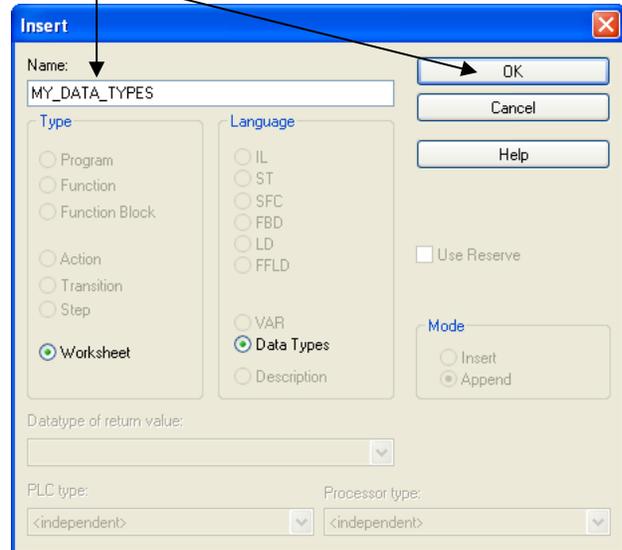
We’re going to describe how to create the arrays for Port 2 only. Port 3 would be similar.

1. To configure the data types for these arrays, first *right-click* on the ‘Data Types’ item in the project tree in ControlWave Designer, and choose **Insert → Data Types** from the pop-up menu.



2. Enter a name for your Data Types worksheet, and click on [OK].

Enter a name for the worksheet, then click "OK".



3. Now we need to define the data types used for the arrays. The array size is determined based on the maximum number of EBSAP slave nodes under a given virtual node on a port, and the number of virtual nodes on the port. Since, in this example, the largest number of EBSAP slave nodes is 90 we want to create an array type called 'MaxSlaves' that consists of 90 BOOL values. Since we happen to have up to two virtual nodes on a port, we want to make another array type called 'NumVirtNodes' that consists of a column of size 'MaxSlaves' for each virtual node under the port.

```

1  TYPE
2      MaxSlaves:      ARRAY [1..90] OF BOOL;
3      NumVirtNodes:  ARRAY [1..2] OF MaxSlaves;
4  END_TYPE

```

Here we define the data types used for the arrays.

Note:

Although we made NumVirtNodes dimensions to be [1..2], we could have it [3..4] based on the local address, since it just needed a dimension of 2.

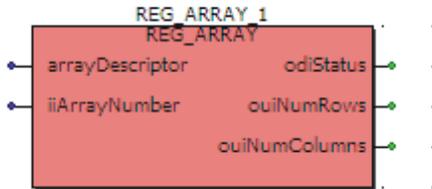
4. Now that the array data types have been defined, we can define the actual array variables that will be used to report on polling success or turn ON/OFF polling for EBSAP slaves on this port. We will call them PORT2_ESLAVE_DEAD, and PORT2_ESLAVE_DISABLE. They must be defined in the Global Variables worksheet (so they are accessible throughout the project) and they will both be of the array type 'NumVirtNodes'.

	Name	Type	Usage
	Default		
	PORT2_ESLAVE_DEAD	NumVirtNodes	VAR_GLOBAL
	PORT2_ESLAVE_DISABLE	NumVirtNodes	VAR_GLOBAL

Insert new variables in the Global Variables worksheet. You must specify them to be of the array data type you defined previously.

5. Configure two REG_ARRAY function blocks, one for each of these arrays:

- The **arrayDescriptor** parameter should be set to the name of the array (either PORT2_ESLAVE_DISABLE or PORT2_ESLAVE_DEAD in this case.)
- The **iiArrayNumber** parameter assigns an array number to the array, so it may be requested by external programs such as OpenBSI's DataView or Harvester.
- The remaining parameters, **odiStatus**, **ouiNumRows**, **ouiNumColumns** need only be assigned variable names; since they are output parameters.



6. On the 'Port Detail' page of the System Variable Wizard for Port 2, you must identify the array number for each of these arrays, so that they can be properly associated with the port. You must check the box next to `_Px_DEAD_ARRAY` and `_Px_DISABLE_ARRAY`; this will create system variables called `_P2_DEAD_ARRAY` and `_P2_DISABLE_ARRAY` (since this is for port 2). The values of these system variables are then set to the same number you assigned to the `iiArrayNumber` parameter in the `REG_ARRAY` function block. In this case, we have specified array numbers 10 and 11.

The screenshot shows the 'Port 1 Configuration' dialog box with the following settings:

Section	Parameter Name	Value
Common	<input type="checkbox"/> Poll Time	<code>_Px_POLL_PER</code> 5
	<input type="checkbox"/> Write Delay	<code>_Px_WRITE_DEL</code> 0
	<input type="checkbox"/> Write (CTS) Timeout	<code>_Px_WRITE_TMO</code> 2500
	<input type="checkbox"/> Ignore Echo Data	<code>_Px_IGNORE_ECHO</code> FALSE
	<input type="checkbox"/> Port Supports Dial	<code>_Px_DIAL_PORT</code> FALSE
	<input type="checkbox"/> Port performs Auto DTR shutdown	<code>_Px_AUTO_DTR</code> FALSE
	<input type="checkbox"/> BSAP Pad Front	<code>_Px_PAD_FRONT</code> 0
	<input type="checkbox"/> BSAP Pad Back	<code>_Px_PAD_BACK</code> 0
Slave	<input type="checkbox"/> Time Sync Disable	<code>_Px_TS_DIS</code> FALSE
	<input type="checkbox"/> Time Sync Needed	<code>_Px_TS_FORCE</code> NO VAL
	<input type="checkbox"/> Node Routing Table Disable	<code>_Px_NRT_DIS</code> FALSE
	<input type="checkbox"/> Alarm Disable	<code>_Px_ALM_DIS</code> FALSE
	<input type="checkbox"/> Immediate Response Disable	<code>_Px_IMM_DIS</code> FALSE
	<input type="checkbox"/> Fast Radio Interval	<code>_Px_CYCLE_INT</code> 0
	<input type="checkbox"/> Fast Radio On Time	<code>_Px_CYCLE_TIMEO</code> 0
	<input type="checkbox"/> Local Port	<code>_Px_LOCAL_PORT</code> FALSE
Master	<input type="checkbox"/> Retries	<code>_Px_RETRIES</code> 3
	<input type="checkbox"/> Data Link Timeout	<code>_Px_TIMEOUT</code> 500
	<input type="checkbox"/> Idle Polling	<code>_Px_IDLE_POLL</code> FALSE
	<input type="checkbox"/> VSAT - Up Ack Wait	<code>_Px_VSAT_UP_ACK_WAIT</code> 0
EBSAP Master	<input type="checkbox"/> Max Slaves Under Virtual Nodes	<code>_Px_MAX_SLAVES</code>
	<input type="checkbox"/> Top Level Nodes	<code>_Px_TOP_LEVEL_NODES</code>
	<input type="checkbox"/> Total Slaves On Port	<code>_Px_TOTAL_NODES</code>
	<input checked="" type="checkbox"/> Array Number of Dead Array	<code>_Px_DEAD_ARRAY</code> 10
	<input checked="" type="checkbox"/> Array Number of Disable Array	<code>_Px_DISABLE_ARRAY</code> 11

At the bottom of the dialog, there is a note: "x = Port Number". Buttons for "OK" and "Cancel" are also visible.

Check these boxes to create the EBSAP system variables for this

These numbers identify the registered array numbers (done through `REG_ARRAY`).

7. Once your project has been compiled and downloaded, these arrays can be used to enable/disable polling for EBSAP slave nodes on Port 2, and to report whether responses are being received from EBSAP slave nodes on that port.
8. Now, let's consider how to use these arrays that we've created. *It may help your understanding if you refer to the figure on page 136 of this section that shows the network.*

Individual array elements of a single-dimension array (`_SLAVE_DEAD` or `_SLAVE_POLL_DIS`) are referenced by their row number, for example `_SLAVE_DEAD[row_number]`. So if you reference `_SLAVE_DEAD[5]` you are referencing the BOOL value in row 5 of array `_SLAVE_DEAD`.

Individual array elements from a two-dimensional array (such as `P2_ESLAVE_DISABLE`) are referenced by row and column number, for example `P2_ESLAVE_DISABLE[row][column]`. So if you reference `P2_ESLAVE_DISABLE[3][17]` references the BOOL value in the row 3 (a virtual node) and column 17 (the 17th EBSAP slave node under that virtual node.)

`_SLAVE_DEAD`

To see whether responses are being received for poll messages from *any* EBSAP slave nodes under the first virtual node on Port 2 (which has a local address of 3), we would look at the BOOL value of `_SLAVE_DEAD[3]`, where 3 is the row number of the array.

To see whether responses are being received for poll messages from *any* EBSAP slave nodes under the second virtual node on Port 2 (which has a local address of 4), we would look at the BOOL value of `_SLAVE_DEAD[4]`.

Note:

By default, A TRUE BOOL value indicates the virtual node is DEAD, i.e. there are no 'live' EBSAP slaves for this virtual node.

`_SLAVE_POLL_DIS`

This is very similar to the `_SLAVE_DEAD` array, except it actually turns polling ON or OFF.

To turn ON/OFF polling for *any* EBSAP slave nodes under the first virtual node on Port 2 (which has a local address of 3), we would manipulate the BOOL value of `_SLAVE_POLL_DIS[3]`.

Note:

Interpretation of the meaning of the BOOL value is determined by the system variable `_BSAP_FLAG_SENSE`.

`PORT2_ESLAVE_DEAD`

We had said, as part of this example, that PORT2, an EBSAP Master Port, had two virtual nodes underneath it. The first virtual node on PORT2 has 90 EBSAP slave nodes; the second virtual node on PORT2 has 72 EBSAP slave nodes.

To find out whether responses have been received from any particular node from among the 90 EBSAP slaves underneath the first virtual node, we would examine the BOOL values of `PORT2_ESLAVE_DEAD[1][1]` to `PORT2_ESLAVE_DEAD[1][90]`.

To find out whether responses have been received from any particular node from among

the 72 EBSAP slaves underneath the second virtual node, we would examine the BOOL values of PORT2_ESLAVE_DEAD[2][1] to PORT2_ESLAVE_DEAD[2][72].

Note:

By default, a TRUE BOOL value in this array indicates that the associated EBSAP slave node is dead. This interpretation can be reversed by `_BSAP_FLAG_SENSE`.

PORT2_ESLAVE_DISABLE

Again, PORT2 is an EBSAP Master Port that has two virtual nodes underneath it. The first virtual node on PORT2 has 90 EBSAP slave nodes; the second virtual node on PORT2 has 72 EBSAP slave nodes.

To enable/disable polling for any particular node from among the 90 EBSAP slaves underneath the first virtual node, we would manipulate the BOOL values of PORT2_ESLAVE_DISABLE[1][1] to PORT2_ESLAVE_DISABLE[1][90].

To enable/disable polling for any particular node from among the 72 EBSAP slaves underneath the second virtual node, we would examine the BOOL values of PORT2_ESLAVE_DISABLE[2][1] to PORT2_ESLAVE_DISABLE[2][72].

Note:

Interpretation of the meaning of the BOOL value is determined by the system variable `_BSAP_FLAG_SENSE`.

Defining the Virtual Nodes

On the network level immediately below the EBSAP Master are one or more virtual nodes. Although, virtual nodes are really just software structures residing in the EBSAP Master, virtual nodes are created in NetView's RTU Wizard, just like any other RTU in your network. The only difference is that you have to specify that it is a virtual node, instead of a real piece of RTU hardware.

There are two ways to define a virtual node:

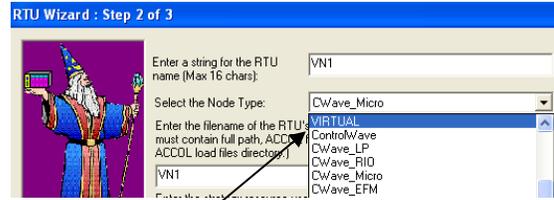
Method 1:

In NetView, drag a Virtual node icon from the Toolbox into your network hierarchy. This will activate the RTU Wizard, from which you can proceed to define the node as you would any other RTU.



Method 2:

If you're already in the RTU Wizard, choose 'VIRTUAL' as the node type and then continue to define it as you would any other RTU.



Choose "VIRTUAL"

Defining the EBSAP Slave Nodes

An EBSAP Slave node can either be a ControlWave-series controller or a Network 3000-series controller. This section will only discuss configuring ControlWave-series controllers as EBSAP slave nodes; for information on using Network 3000-series controllers as EBSAP slave nodes, please consult the *Expanded Node Addressing* section of the *ACCOL II Reference Manual* (document# D4044).

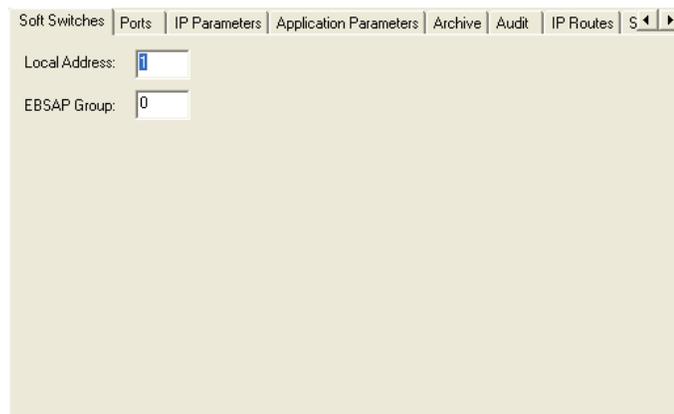
Defining the EBSAP Slave Port

Each EBSAP slave node must have an EBSAP Slave Port configured via the Flash Configuration Utility. (NOTE: A Slave Port could be used, instead, however, we recommend EBSAP Slave Ports to ensure proper communication statistics are collected.)

Specifying the EBSAP Group Number for a Slave Node

All RTU's immediately below the *first* virtual node on a given EBSAP communication line belong to group 0; all RTU's immediately below the *second* virtual node on a given EBSAP communication line belong to group 1, *and so on*.

Because of this, if you have multiple EBSAP communication lines from a particular EBSAP Master node, each EBSAP line will have a group 0 for its first virtual node, a group 1 for its second virtual node, etc., depending upon how many virtual nodes are on that line.



For the current generation of ControlWave RTU's, the group number of a particular RTU is assigned in the "EBSAP Group" field of the 'Soft Switches' page of the Flash Configuration Utility.

Note:

See *Expanded Node Addressing* in the *ACCOL II Reference Manual* (document# D4044) if you have an older Network 3000 RTU which cannot store a group number in FLASH memory.

If you are NOT using EBSAP in your network, this must be left at the default of '0'.

Use the [**Save to Rtu**] button to save the entries to the controller.

Note:

Changes to soft switches do not take effect until the RTU has been reset.

Example 1 – OpenBSI Workstation is EBSAP Master to 1000 ControlWave controllers

Suppose there is an OpenBSI Workstation, and 3 ports which will be defined as EBSAP Master lines. Two of the EBSAP lines need to communicate with 400 ControlWave controllers, and a third EBSAP line needs to communicate with 200 ControlWave controllers, for a total of 1000 ControlWave controllers in the network. Here's how to approach this.

1. In NetView, define 3 separate EBSAP lines (COM1, COM2, COM3). COM1 and COM2 will each communicate with 400 RTUs, and COM3 will communicate with 200 RTUs.
2. On level 1 of the network, define virtual nodes and on level 2 of the network define the real 1000 RTUs.

For COM1, 4 virtual nodes must be defined in order to address 400 real RTUs.

VN1, VN2, VN3, and VN4 are added in NetView on Level 1, by selecting 'VIRTUAL' as the node type. (By the way you don't have to name the virtual node 'VN', we're just doing that for ease of explanation.)

- VN1 will have a local address of 1, and it is responsible for RTU1 to RTU127. RTU1 to RTU127 are in Group 0 because VN1 is the first virtual node on COM1. The local addresses for these RTUs will be 1 to 127. *NOTE: In this example we are assigning real nodes to local address 127, however, certain older RTU types reserve address 127 for special purposes (e.g. redundant DPC 3330s with RASCL) and don't assign an RTU to it. Keep this in mind if you are working with older RTUs.*
- VN2 will have a local address of 2, and it is responsible for RTU128 to RTU254. RTU128 to RTU254 are in Group 1 because VN2 is the second virtual node on COM1. The local addresses for these RTUs will be 1 to 127.
- VN3 will have a local address of 3, and it is responsible for RTU255 to RTU381. RTU255 to RTU381 are in Group 2 because VN3 is the third virtual node on COM1. The local addresses for these RTUs will be 1 to 127.

- VN4 will have a local address of 4, and it is responsible for RTU382 to RTU400. RTU382 to RTU400 are in GROUP 3 because VN4 is the fourth virtual node on COM1. The local addresses for these RTUs will be 1 to 19.
- RTUs 1 to 400 must all be on Level 2 of the Network.

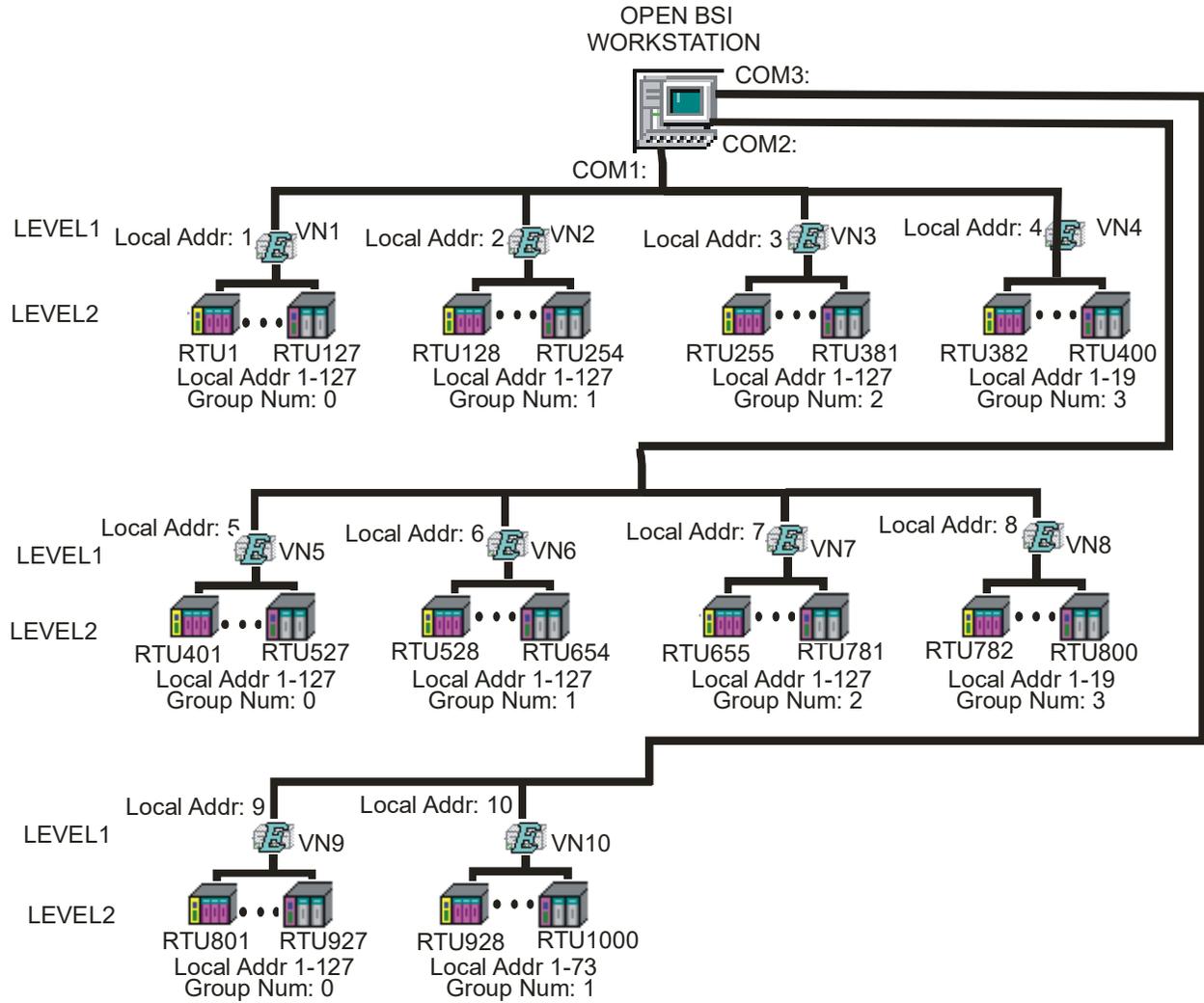
For COM2 we again need to define at least 4 virtual nodes (VN) in order to address another 400 real RTUs. VN5, VN6, VN7, and VN8 are added in NetView, again, by selecting 'VIRTUAL' as the node type.

- VN5 will have a local address of 5, and it is responsible for RTU401 to RTU527. RTU401 to RTU527 are in Group 0 because VN5 is the first virtual node on COM2. The local addresses for these RTUs will be 1 to 127.
- VN6 will have a local address of 6, and it is responsible for RTU528 to RTU654. RTU528 to RTU654 are in Group 1 because VN6 is the second virtual node on COM2. The local addresses for these RTUs will be 1 to 127.
- VN7 will have a local address of 7, and it is responsible for RTU655 to RTU781. RTU655 to RTU781 are in Group 2 because VN7 is the third virtual node on COM2. The local addresses for these RTUs will be 1 to 127.
- VN8 will have a local address of 8, and it is responsible for RTU782 to RTU800. RTU782 to RTU800 are in Group 3 because VN8 is the fourth virtual node on COM2. The local addresses for these RTUs will be 1 to 19.
- RTUs 401 to 800 must all be on Level 2 of the Network.

For COM3 I again need to define at least 2 virtual nodes (VN) in order to address the remaining 200 real RTUs. VN9 and VN10 are added in NetView, on Level 1 again, by selecting 'VIRTUAL' as the node type.

- VN9 will have a local address of 9, and it is responsible for RTU801 to RTU927. RTU801 to RTU927 are in Group 0 because VN9 is the first virtual node on COM3. The local addresses for these RTUs will be 1 to 127.
- VN10 will have a local address of 10, and it is responsible for RTU928 to RTU1000. RTU928 to RTU1000 are in Group 1 because VN10 is the second virtual node on COM3. The local addresses for these RTUs will be 1 to 73.
- RTUs 801 to 1000 must all be on Level 2 of the Network.

The figure on the next page shows the completed network, though there is only space to show the first and last RTU of each group.



Example 2 – ControlWave Controller is EBSAP Master to 300 ControlWave EBSAP Slaves

There are many possible ways to do this with regard to number of ports and virtual nodes. At *least* three virtual nodes would be required beneath the EBSAP Master controller, since no one node can support more than 127 nodes. That could mean, for example, 127 nodes underneath the first virtual node, 127 nodes underneath the second virtual node, and 46 nodes underneath the third virtual node, for a total of 300 EBSAP slaves. Or, we could have 100 nodes underneath each of the three virtual nodes. We could have all three virtual nodes on a single EBSAP Master port, or use more than one EBSAP Master Port. We could even have more than 3 virtual nodes to have a smaller number of EBSAP slaves through each virtual node, but provide more room for expansion. It's up to you how you want to do it.

For this example let's have 3 virtual nodes, each with 100 EBSAP Slave nodes, and we'll put them all on the same port (Port 2). The virtual nodes use local addresses 3 for and 5.

Here's how we configure this.

1. Using the OpenBSI Flash Configuration Utility, configure an EBSAP Master Port for the ControlWave controller which will serve as the EBSAP Master. The **"Low Virtual Slave"** must be set to 3 and the **"High Virtual Slave"** must be set to 5. Since we will have no more than 100 EBSAP slaves under each one, make the **"Max Slaves"** 100.
2. Configure the Control and Status Arrays.
 - Because they are present in all projects by default, you can configure the `_SLAVE_DEAD` and `_SLAVE_POLL_DIS` arrays by just checking the **"Master – Dead_Slaves"** and **"Don't Poll Array"** boxes on the 'Port Globals' page of the System Variable Wizard. Then use two `REG_ARRAY` function blocks (one for each array) to register the arrays.

Note:

The next arrays are optional, but we strongly recommend you create them:

- Under the 'Data Types' item in the ControlWave Designer project tree, add a worksheet to define new data types. You will need to create something like this:

TYPE

MaxSlaves: ARRAY [1..100] OF BOOL;

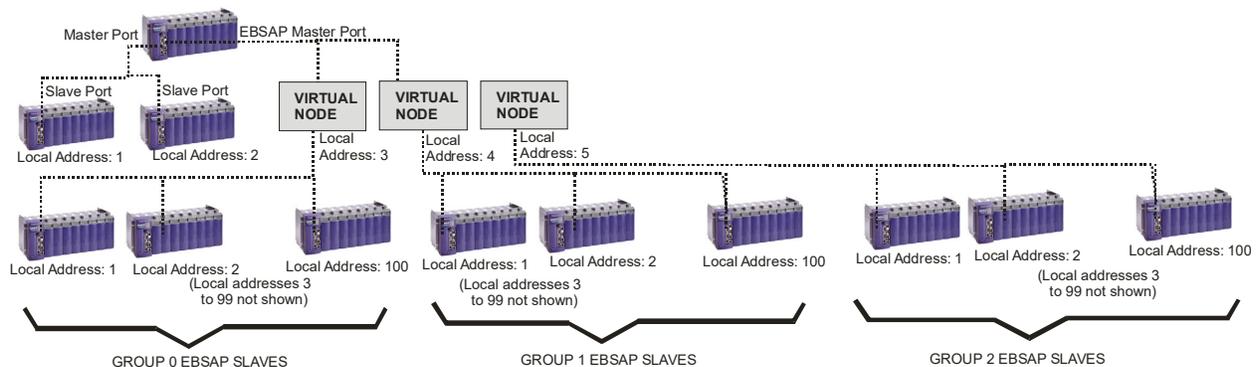
NumVirtNodes: ARRAY [3..5] OF MaxSlaves;

END_TYPE

Note: The numbers used to dimension the NumVirtNodes data type can be different, so long as the size is correct. For example, [1..3] could be used since there are three virtual

nodes; [3..5] could be used since their local addresses are 3,4 and 5, or you could use [0..2] since that would represent the EBSAP group numbers (0, 1, and 2) of their respective EBSAP Slaves. It doesn't matter so long as the size of 3 is correct. Note also that the names 'MaxSlaves' and 'NumVirtNodes' can be changed; those are just the names we chose.

- In the Global Variables worksheet, define two arrays of type 'NumVirtNodes' (or whatever you called that type). One should be used for handling reporting of polling success for Port 2 EBSAP slaves; the other should be used for enabling/disabling polling of individual EBSAP slaves on Port 2. Assign numbers to these arrays using REG_ARRAY function blocks.
 - In the System Variable Wizard, go to the 'Port Detail' page for Port 2. You must check the box next to _Px_DEAD_ARRAY and _Px_DISABLE_ARRAY; this will create system variables called _P2_DEAD_ARRAY and _P2_DISABLE_ARRAY (since this is for port 2). The values of these system variables are then set to the same numbers you assigned to the **iiArrayNumber** parameters in the REG_ARRAY function blocks.
3. Define three virtual nodes in NetView's RTU Wizard. They would have local addresses 3, 4, and 5, and would be defined like any other node, except the type is 'VIRTUAL'.
 4. Configure the EBSAP Slave nodes in NetView As we've said, there are to be 300 of them; 100 under each virtual node. In the OpenBSI Flash Configuration Utility, assign the proper group number for each EBSAP Slave. The nodes underneath the *first* virtual node must be assigned to Group 0; the nodes underneath the *second* virtual node must be assigned to Group 1, and the nodes underneath the *third* virtual node must be assigned to Group 2. Each EBSAP Slave node must also be configured with an EBSAP Slave Port.

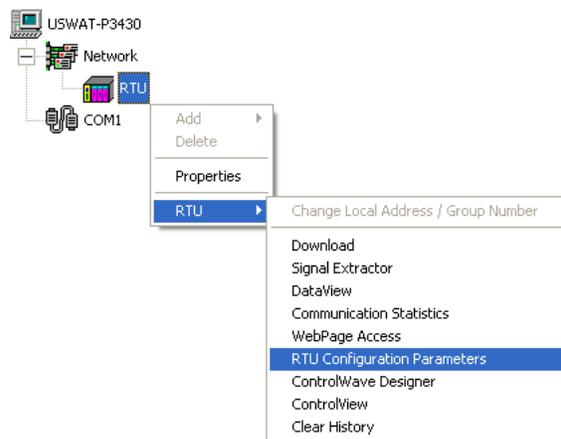


Flash Configuration Utility – An Overview

Use the Flash Configuration Utility to set various configuration parameters in the ControlWave-series controller. Some changes to flash parameters only take effect after the unit has been powered down and then back on. In addition, changes you make to some flash parameters are **not** activated unless the controller's default switch is in the ON position. For a ControlWave, the default switch is SW1-3; for ControlWave LP, the default switch is SW4-3; and for the ControlWave MICRO, the default switch is SW2-3.

Starting the Flash Configuration Utility

The Flash Configuration Utility is initially activated via 'Configure' mode in LocalView. Once LocalView has been used to set these parameters, and other system configuration has been completed, the parameters may be subsequently modified by calling up the same utility from within NetView (or LocalView) by clicking on the icon for the RTU, and then pressing the right mouse button and choosing **RTU → RTU Configuration Parameters**.



The various configuration settings are separated into different pages of the utility. You can access them by clicking on the tab for a particular page.

The screenshot shows the 'Flash Configuration - RTU' window. At the top, there are tabs: 'Soft Switches', 'Ports', 'IP Parameters', 'Application Parameters', 'Archive', 'Audit', 'IP Routes', and 'S'. The 'Soft Switches' tab is active, showing 'Local Address' (1) and 'EBSAP Group' (0). On the right side, there is a vertical stack of buttons: 'Apply New Node', 'Sign On', 'Read From RTU', 'Write To RTU', 'Read From FCP', 'Write To FCP', 'Read From NDF', 'Write To NDF', 'Close', and 'Help'. A status bar at the bottom left reads 'Status: Data Loaded from RTU'. Callout boxes provide the following explanations:

- Click on any of these tabs to bring up other pages of the Flash Configuration utility.** (Points to the tabs)
- This is only useful when using NetView. It allows you to close the session with the current controller, while still leaving the current values on the various pages of the utility. This allows you to configure a different controller, without having to re-enter values in all the fields.** (Points to the 'Sign On' button)
- You must click here to sign-on with a username and password in order to access any flash parameters.** (Points to the 'Sign On' button)
- This button reads the current configuration from the controller into the utility.** (Points to the 'Read From RTU' button)
- This button saves ALL changes to the controller.** (Points to the 'Write To RTU' button)
- This button reads the current configuration from the Flash Configuration (FCP) file.** (Points to the 'Read From FCP' button)
- This button saves ALL changes to the FCP file.** (Points to the 'Write To FCP' button)
- This button reads the current configuration from the NETDEF files into the utility.** (Points to the 'Read From NDF' button)
- This button saves ALL changes to the NETDEF files.** (Points to the 'Write To NDF' button)
- This button shuts down the Flash Configuration utility.** (Points to the 'Close' button)

The various pages of the utility are discussed, briefly, below:

Soft Switches

This page allows you to set the BSAP local address of the controller, and, if using expanded node addressing, the EBSAP group number.

Ports

This page allows configuration of all communication ports on the controller: serial BSAP ports, serial IP ports (PPP), and Ethernet IP ports.

IP Parameters

This page allows you to set certain parameters for IP communications such as the IP address of the Network Host PC (NHP), UDP socket numbers, and the address of the default gateway.

Application Parameters

This page allows you to set 'tuning' parameters which govern how the ControlWave executes its application (project).

Archive

Archive data is one portion of the historical capabilities of the ControlWave-series controller. It allows 'snapshots' of many variables to be saved at the same instant, to provide a detailed historical record of process variables at a particular moment in time. The archive data is saved at the controller, in structures called **archive files** and is configured, in part, using the ARCHIVE function block in your ControlWave project. Archive files may be collected by OpenBSI Utilities such as DataView, or the Harvester.

Audit

Audit Trail is one portion of the historical capabilities of the ControlWave controller. It allows records to be kept of when certain variables change value, as well as recording all alarms in the system. The Audit page specifies various parameters used to set up the Audit Trail system. Configuration is also performed, in part, using the AUDIT function block in your ControlWave project.

IP Routes

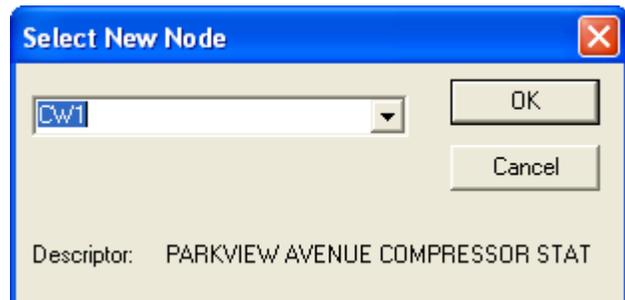
Dynamic IP routes allow messages which cannot successfully reach a particular destination address, to be re-routed through a different path in the IP network.

Security

This page allows configuration of user accounts and privileges.

Push Buttons:

[Apply New Node] This button is only useful when the Flash Configuration utility is started from within NetView (since no other nodes are accessible in the Select New Node dialog box within LocalView).

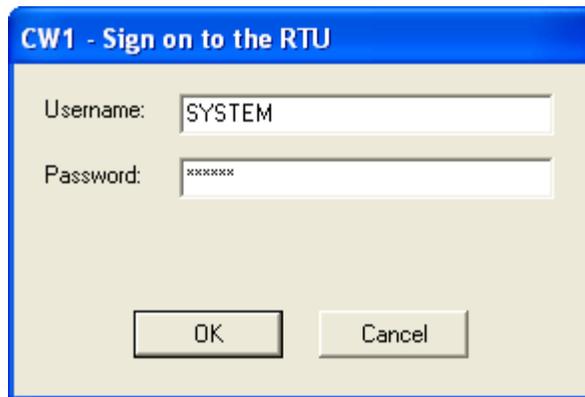


It allows the session with the current controller to be closed, and then allows the user to select a *different* controller for configuration, without reinitializing the values in the pages of the utility.

The new controller must have been defined within the NETDEF files.

One application of this is to open a session with a new node, and then load configuration information from the NETDEF file(s) that was for a *different* node (via **[Read From NDF]**). This can be useful if multiple nodes have similar configurations; the common configuration can be brought into the utility, and then the unique portions only need to be modified for each individual controller.

[Sign On] - This button must be used to sign-on to the controller with a username and password prior to reading or writing Flash parameters.



Note:

If you do NOT sign on, the first time you attempt a read/write operation with the controller, you will be prevented from doing so, and will be prompted to sign on then.

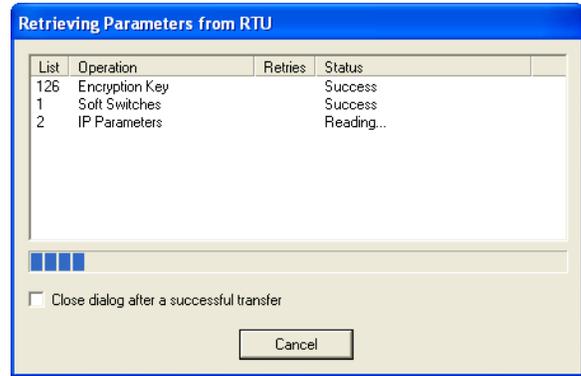
[Read From NDF] - This button reads the current configuration of this controller as specified in NetView's NETDEF files, and copies it into the pages of the Flash Configuration Utility. This can be particularly useful in a situation where the CPU board of a controller has failed, and the replacement board must be configured; this allows the configuration to be called up from the NETDEF, and subsequently copied into the controller using the **[Write To RTU]** button. NOTE: This operation can only be performed from within NetView, or when you start LocalView in Configure Mode.

Note

The reason other LocalView modes (such as Local or Flash) cannot perform these operations is that **only** the Configure mode allows you to specify a particular NETDEF file for modification (by checking the “Use an Existing Configuration (.ndf) File” and then by identifying the path and name of the NETDEF). The other modes use a *temporary* NETDEF which disappears on program exit.

[Write To NDF] This button causes all entries made in the Flash Configuration Utility for the current controller to be copied into the current NETDEF file. This avoids the need to re-enter the same configuration information in NetView. This operation will only work when the Flash Configuration Utility is invoked from within NetView or when LocalView is in Configure Mode; otherwise a permanent NETDEF file is not available to write to (see note above).

[Read From RTU] - This button reads the current configuration characteristics directly from the controller, and copies them into the pages of the Flash Configuration Utility. These can subsequently be stored in the NETDEF using the **[Write To NDF]** button (see above), to avoid the need to re-enter the same configuration details inside the NetView program. **Note:** If you haven't signed on prior to clicking on this button, you will be prompted to do so.



[Write To RTU] - This button saves ALL entries in the pages of the Flash Configuration Utility to the ControlWave-series controller. **NOTE:** If you haven't signed on prior to clicking on this button, you will be prompted to do so.



After the write operation completes, the Flash Configuration Utility prompts you to reset the RTU, if a reset is needed.

[Read FCP] - This button reads the current configuration of this controller, as specified in a Flash Configuration Profile file (*.FCP), and copies it into the pages of the Flash Configuration Utility. The flash configuration can subsequently be copied into the controller using the **[Write To RTU]** button.

Note:

We recommend that you never edit the file FCP manually because no validation is performed on the file when the utility opens it. Improper edits could corrupt the file.

[Write FCP] - This button causes all entries made in the Flash Configuration Utility for the current controller to be copied into the Flash Configuration Profile file (*.FCP).

[Close] This button shuts down the Flash Configuration Utility.

For details on the individual pages of the Flash Configuration Utility, see *Chapter 5* of the *OpenBSI Utilities Manual* (part number D301414X012).

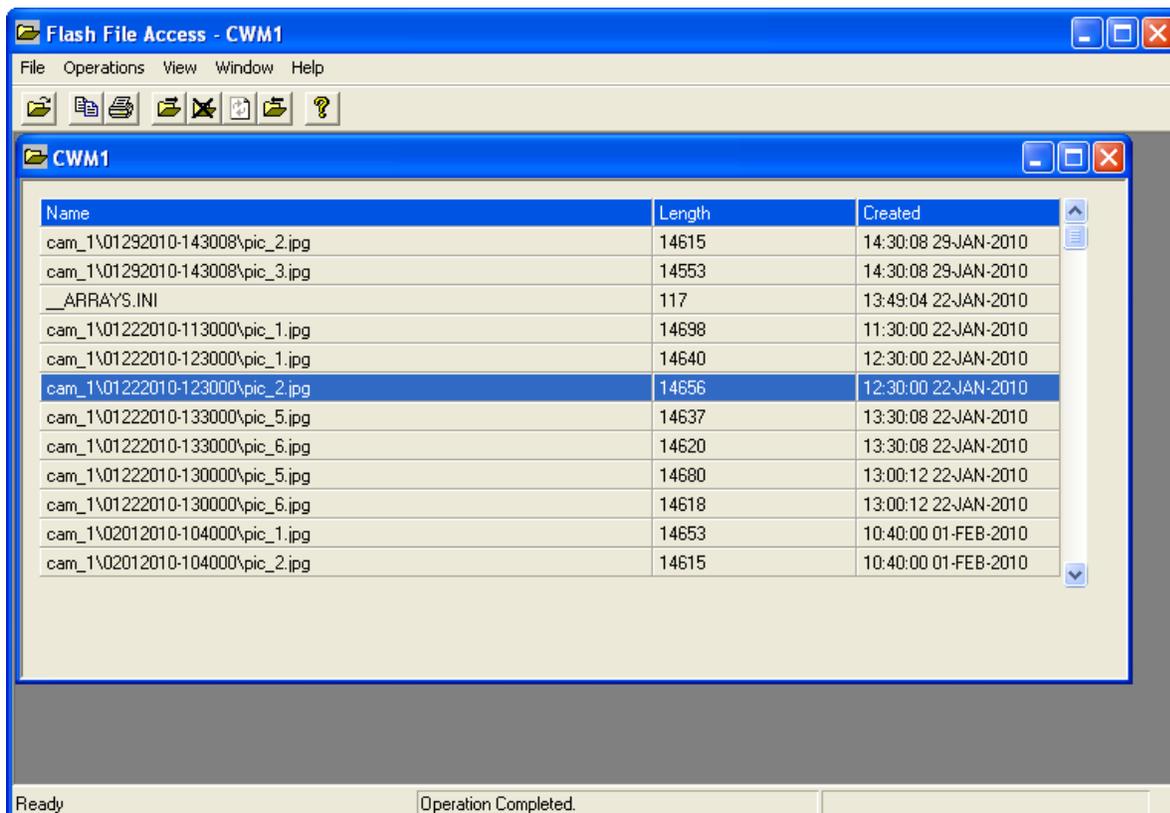
Flash File Access

The Flash File Access utility lets you view a list of user files residing in the ControlWave's flash area. You can optionally delete files from the flash area, upload files from the ControlWave flash to the OpenBSI workstation, or send files from the OpenBSI workstation to the flash area.

Important

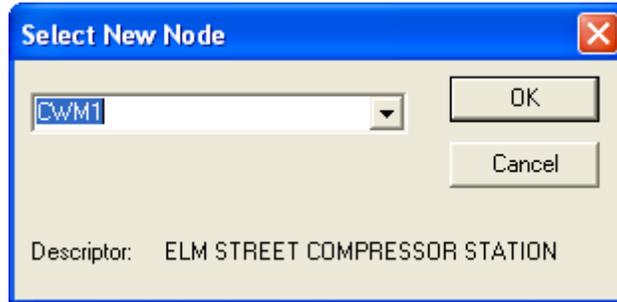
This utility reads/writes files to and from the ControlWave's flash file memory area, but it cannot be used to download ControlWave system firmware or to send/retrieve historical files (audit/archive). Although those items may reside in flash, the Flash File Access utility does not access those portions of flash. Flash file access is only appropriate for downloading, uploading, or deleting user files such as ControlWave zipped source files (*.zwt), web pages, etc. For example, a typical usage would be to delete excess files to free up flash space.

With communications active in LocalView or NetView, click **Start→Programs→OpenBSI Tools→Debugging Tools→ Flash File Access (FileDirect)**



Viewing a List of Files in the Flash File Area:

1. Either click **File** → **New RTU** or click the New RTU icon .
2. In the Select New Node dialog box, select the RTU with which you want to communicate and click **OK**.

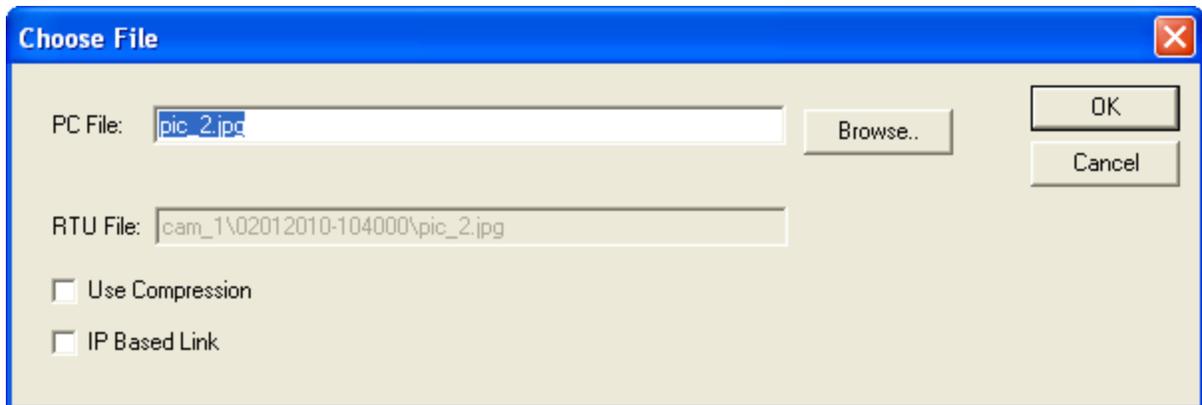


3. The utility generates a list of files in the user flash file area. It may take some time for it to collect all the file details.

Uploading a File from the ControlWave to Your OpenBSI Workstation:

With the list of files in the user flash files area visible, do the following:

1. Click on the file you want to upload to the PC, so it is highlighted.
2. Either click **Operations** → **Upload File** or click the Upload File icon . The Choose File dialog box opens.

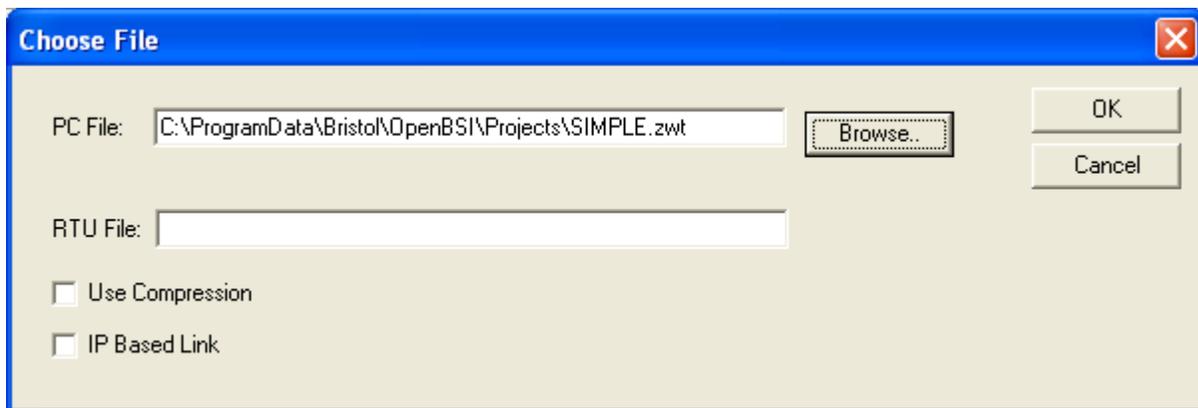


3. In the **PC File** field of the Choose File dialog box, you can optionally specify a new name for the file on the PC, and select a destination folder on the PC for the file, using the **Browse** button.
4. If you want the utility to compress the file during transfer, select **Use Compression**. (Note: Do not use compression for JPG or ZWT files which are already compressed.)

5. If your communication with the ControlWave is via IP, select **IP Based Link**. This allows for faster file transfers on IP links.
6. Click **OK** to initiate the upload, and the utility uploads the file to the specified folder on your PC.

Copying a File from the OpenBSI Workstation to Your ControlWave:

1. Either click **Operations** → **Copy to RTU..** or click the Copy File icon . The Choose File dialog box opens.
2. Use the **Browse** button to specify the path and filename of the PC you want to copy to the ControlWave user flash files area.



3. In the **RTU File** field, you can optionally specify a new name for the file at the ControlWave.
4. If you want the utility to compress the file during transfer, select **Use Compression**. (Note: Do not use compression for JPG or ZWT files which are already compressed.)
5. If your communication with the ControlWave is via IP, select **IP Based Link**. This allows for faster file transfers on IP links.
6. Click **OK** to initiate the copy, and the utility copies the file from the OpenBSI workstation to the ControlWave.

Deleting a File from the ControlWave User Flash Files Area:

With the list of files in the user flash files area visible, do the following:

1. Click on the file you want to delete, so it is highlighted.
2. *Be certain this is the file you want to delete, because there is no undo-delete or prompting to confirm the deletion.*
3. Click **Operations** → **Delete** or click on the Delete icon. 

4. The utility erases the specified file.

Refreshing the List of Files:

Either click **Operations** → **Refresh** or click on the Refresh icon. 

Function Blocks – Creating

Suppose we have created a program in ControlWave Designer that we want to re-use. For example, let's say we created a program that controls the flow of liquid in a pipeline. (To see a simple example of such a program, refer to the *Getting Started with ControlWave Designer Manual*, part number D301416X012.)

Now, however, we need to perform the same exact flow control operation on eight different pipelines using a single ControlWave controller. Let's also assume that the initial values for all of the different parameters for the re-used program (with the exceptions of the input, output, and setpoint) will be the same for each of the eight pipelines. There are several different ways we could do that. We *could* create seven additional program POUs for the project, and repeat the exact same steps we did to create the first program, except we would create multiple program instances of that same program. That would be somewhat tedious.

Another approach would be to go back to our original flow control program, and use the **Edit→Copy** and **Edit→Paste** commands to copy multiple sets of the function blocks around in the same worksheet, and then change the variables for each one. That would be a little quicker, but still tedious.

A third solution, which we will discuss here, is to create a user defined function block from our original program. You probably noticed when creating your first program in ControlWave Designer that you had the option of defining a variable as either *local* or *global*. Local variables are only accessible within the current POU (that is, a function, function block, or program). If you define a variable as a *local* variable, and you create *another* POU, the local variables in the first POU are completely unknown to the second POU.

This can be an advantage because the first POU can then be treated as a re-usable little sub-routine which performs some sort of calculation or function. The values of variables are passed into the sub-routine as parameters. The sub-routine uses the values, and performs its calculations locally, inside the sub-routine, and then it passes out an answer.

A user defined function block is such a sub-routine. It is made up of other functions and function blocks. To the user, the whole user-defined function block is like a *black box*. You send inputs into it, any local calculations are performed inside, out of view, and you get outputs from it.

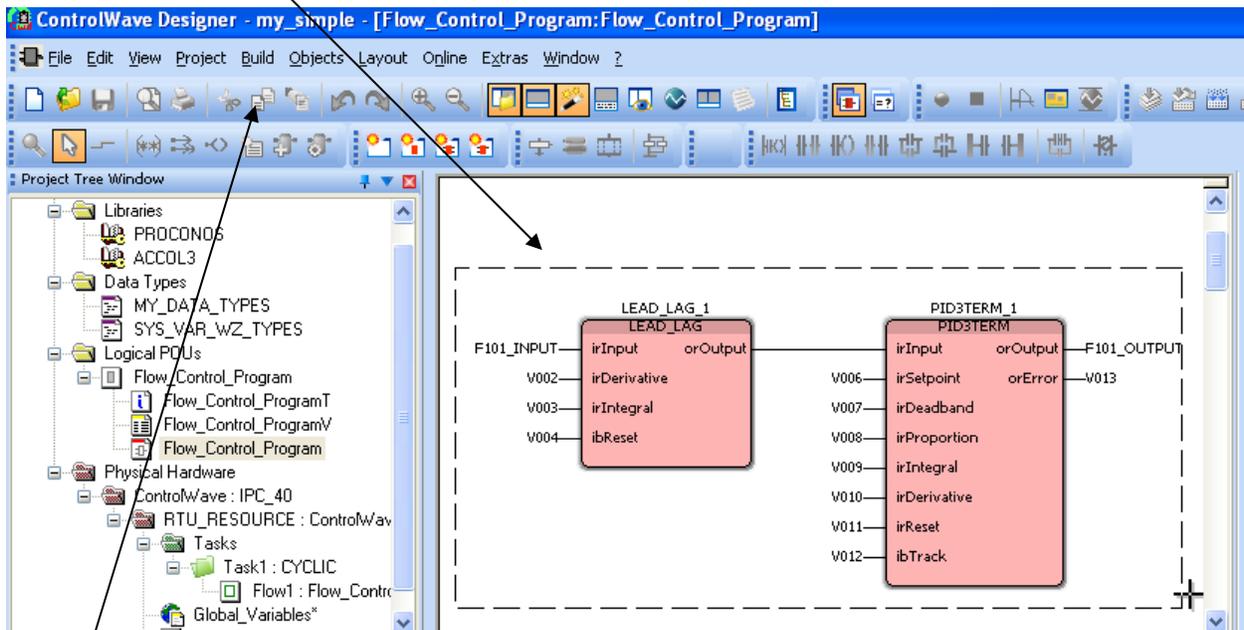
Let's look at the case of the flow control program we discussed previously. Typically, once such a program has been tested, and proper initial values defined, many of the parameters probably won't be changed. The input and output process variables, though, will have to be replaced with other inputs and outputs, based on whichever pipeline you are controlling, and the setup parameter would change, because the setpoint is determined outside of the program, perhaps by an operator. Once you get the other parameters configured the way you want them, though, you might not ever want to change them again. If that is the case, we can create a user-defined function block which can be re-used as many times as you want, and all you need to do is specify the parameters which will

change (for example, an input, an output, and a setpoint for each instance of the user defined function block).

To illustrate this technique, open the project containing the program we want to re-use.

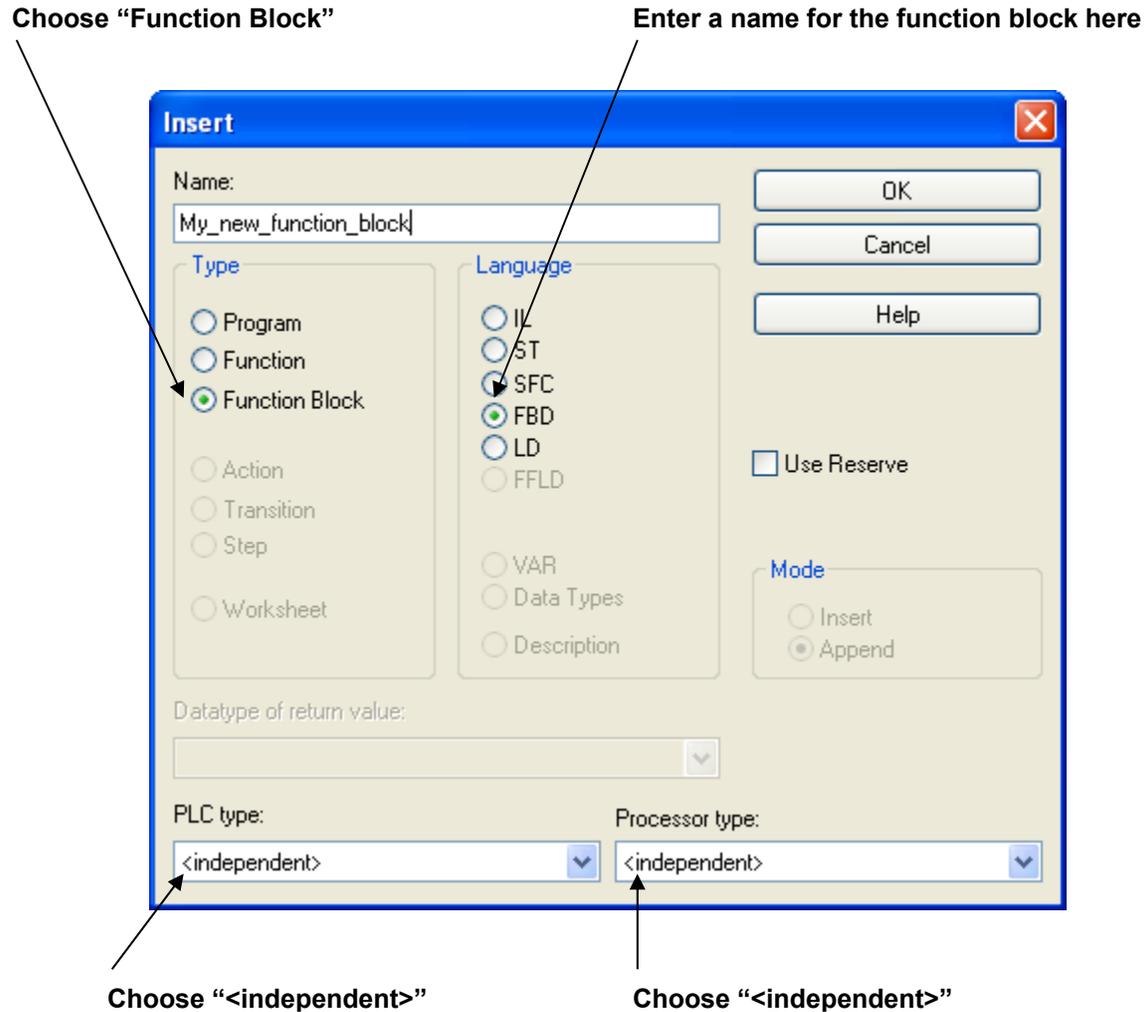
Drag a box around the items in the worksheet that you want to re-use, and when you release the mouse, they will be selected. Click on the 'Copy' icon, or choose **Edit→Copy** from the menu bar.

First, drag a box around the program you want to re-use, then release the mouse to select the items.



Next, click the "Copy" icon, or click Edit → Copy from the menu bar.

Now, we need to create an empty function block to paste our program into, thereby creating a new user-defined function block. To do this, right-click on the Logical POU's section of the project tree and choose **Insert→Function Block** from the pop-up menu.

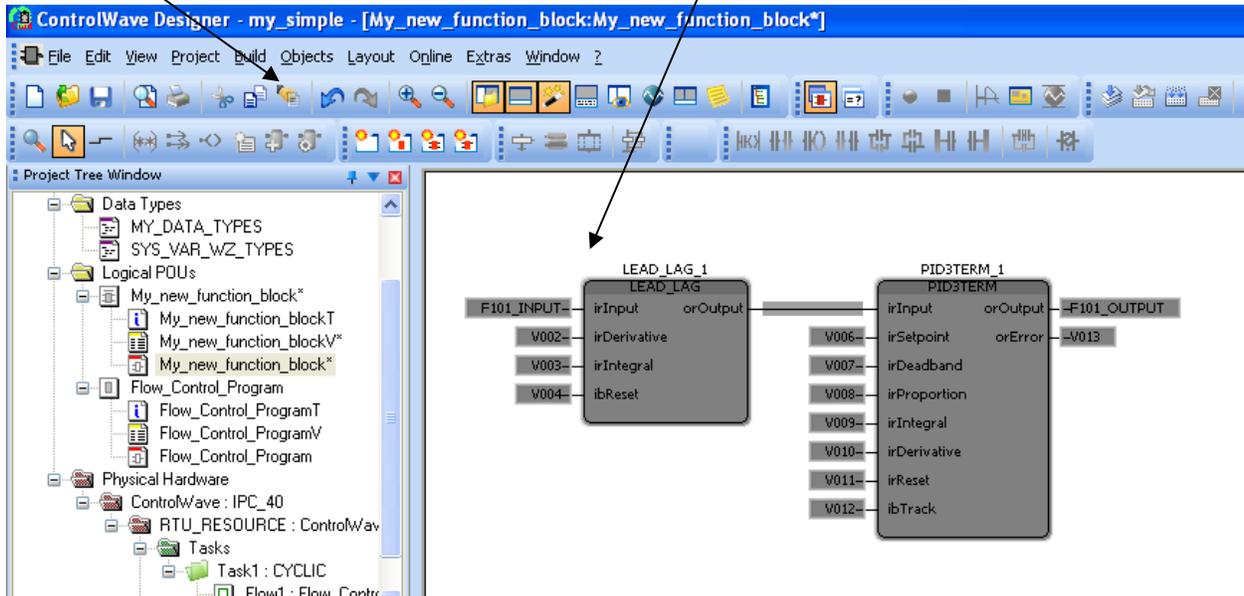


In the Insert dialog box, choose **"Function Block"** for the **"Type"** and enter a name for the user defined function block. (Here we entered 'My_new_Function_Block', but you'll probably want to choose something shorter, and more descriptive.) Specify the PLC type and Processor type as shown, above, and click on **[OK]**. Icons for 'My_new_Function_Block' will be added to a new branch in the project tree.

In the new branch of the project tree, double-click on the third icon, 'My_new_Function_Block*' and an empty worksheet will appear. Click on the 'Paste' icon, or click on **Edit→Paste** and an outline image of the copied program will appear; position the image where you want it in the worksheet and click; the program will be copied into the worksheet.

First, click on the "Paste" icon or click Edit → Paste.

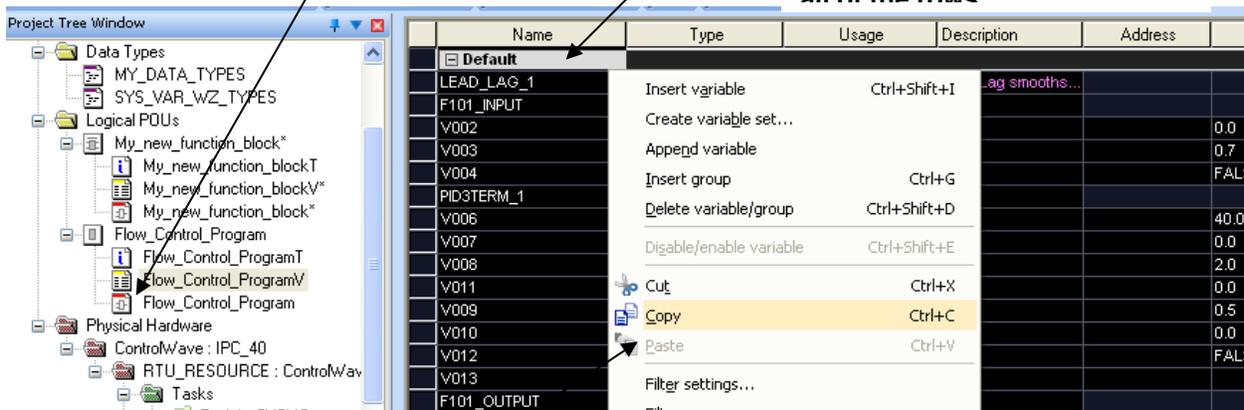
Then position the image where you want it, and click.



Now, copy the variables from the program into your new function block and re-define the variables as necessary.

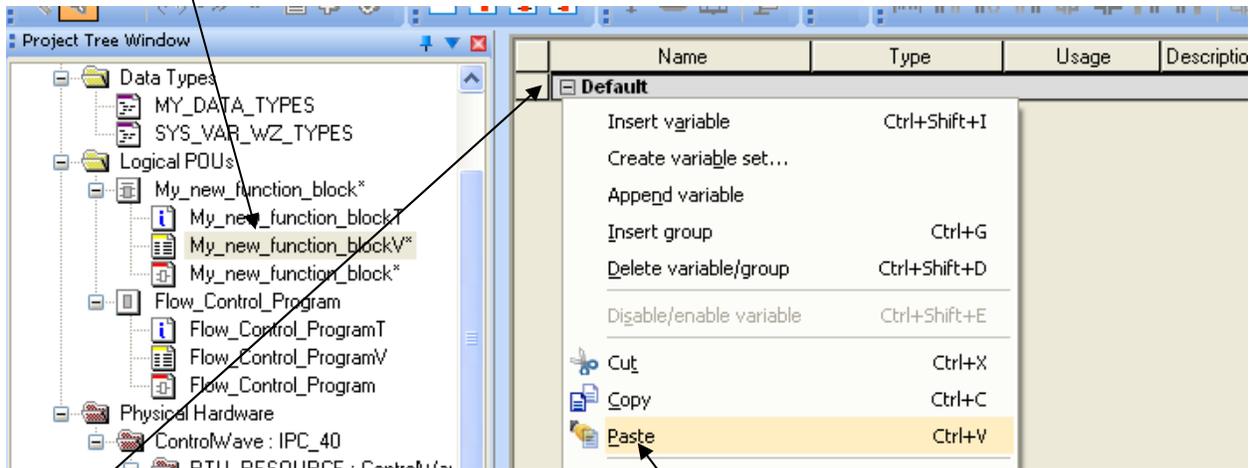
First, double-click on the variables section of the program from which you want to copy.

Second, highlight the entire contents of the variables section. (You can do this by choosing Edit → Select All, or by dragging the mouse over all of the rows)



Finally, right click and choose "Copy" from the pop-up menu.

Now, double-click on the variable section of the new function block you are creating.



When the empty variable window opens, you must click on this box in the upper left part of the window (on the same line as the "Default").

Finally, you can right-click, and choose "Paste" from the pop-up menu.

Now modify the variables section of the new function block.

Any variable which you want to be changeable in the new function block should be made either a VAR_INPUT (if it is an input) or a VAR_OUTPUT (if it is an output). Click in the 'Usage' column to change this.

You will also want to change the names to make them more *generic*: instead of *F101_INPUT* and *F101_OUTPUT* (as in the original program), name them just INPUT and OUTPUT, respectively. The variable used for the setpoint, would be named SETPOINT, with a usage type of VAR_INPUT. You can change these names by clicking in the 'Name' column and typing in the changes.

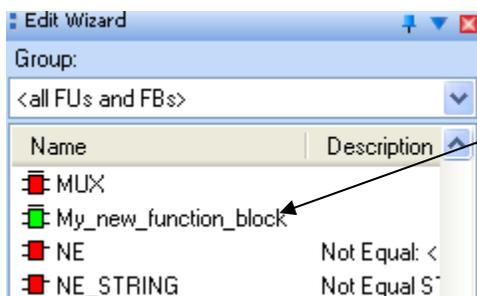
Name	Usage	Description	Ad
Default			
LEAD_LAG_1	VAR	Lead Lag smooths out fluctuations in the input value	
INPUT	VAR_INPUT		
V002	VAR		
V003	VAR		
V004	VAR		
PID3TERM_1	VAR		
SETPOINT	VAR_INPUT		
V007	VAR		
V008	VAR_EXTERNAL		
V011	VAR_EXTERNAL_FB		
V009	VAR_EXTERNAL_PG		
V010	VAR_IN_OUT		
V012	VAR_INPUT		
V013	VAR		
OUTPUT	VAR_OUTPUT		
Default			

To change a variable's name, click on the variable in the "Name" column and make the change.

To change a variable's usage, click in the "Usage" column for the variable, then select the new usage, e.g. VAR_INPUT.

When you have finished making these modifications, save your project, and execute a build.

Now, when you are in the Edit Wizard, if you call up a list of all functions and function blocks via the '<all FUs and FBs>' group, you will see the name of the function block you created.



In the Edit Wizard, when you review the list of all function blocks, you can see the function block you just created in the list.

If you insert the 'My_new_Function_block' function block into one of the programs of your project, it will only show those variables declared as VAR_INPUT and VAR_OUTPUT; the others are hidden. You can then proceed to configure it like any other function block.



Once you have saved your project with your new user-defined function block, you can use it in other projects by importing the first project as a user library. See Libraries later in this manual.

Function Block Parameter Name Prefixes

The following table summarizes the meaning of the letters in the parameter name prefix of function blocks in the ACCOL3 library.

Parameter Name Prefix	Input or Output	Valid Data Types:
ia, iany	INPUT	REAL, SINT, INT, DINT NOTE: You CANNOT use constants on parameters with the 'ia' or 'iany' prefix; only variables may be used.
iab	INPUT	BOOL variable - NO constants allowed.
iais	INPUT	STRING or INT variable. No constants.
iar	INPUT	REAL variable - NO constants allowed.
iarb	INPUT	REAL or BOOL variable. NO constants allowed.
iaus	INPUT	USINT variable or array of USINT. No constants.
ib	INPUT	BOOL variable or constant
idi	INPUT	DINT variable or constant
ii	INPUT	INT variable or constant
ioab	INPUT & OUTPUT	BOOL variable - NO constants allowed.
ioar	INPUT & OUTPUT	REAL variable - NO constants allowed.
ir	INPUT	REAL variable or constant
is, isi	INPUT	SINT variable or constant
is, istr	INPUT	STRING (must be surrounded by single quotes)
iudi	INPUT	UDINT variable or constant
iui	INPUT	UINT variable or constant
ius	INPUT	USINT variable or constant
ob	OUTPUT	BOOL variable or constant
odi	OUTPUT	DINT variable or constant
oi	OUTPUT	INT variable or constant
or	OUTPUT	REAL variable or constant
oud	OUTPUT	UDINT variable or constant
oui	OUTPUT	UINT variable or constant
Ous	OUTPUT	USINT variable or constant

Historical Data

The data in your ControlWave controller is constantly being updated, with the latest available data values collected from field instrumentation (pressure transmitters, temperature transmitters, electrical contacts, switches, etc.) Data from a particular instant in time, however, is only maintained for a short period (usually, no longer than a few seconds, depending upon how fast data is collected in your system) because when a new value is collected, it automatically overwrites the previous value.

Although this is ideal for reporting the current state of process variables in your system, most users need to retain certain data for a longer period of time (hours, weeks, etc.). This data is referred to as **historical data**.

Depending upon your system requirements, historical data might be saved by whatever (graphical user interface (GUI) software is running at your central computer (such as Emerson's OpenEnterprise™, Iconics Genesis®, Intellution® FIX®, or Wonderware®).

Some categories of historical data can also be saved within the controller itself. That is the subject we will address in this section.

What is Historical Data Used For?

Historical data is typically used in printed reports or spreadsheets. Often, records of certain variables such as flow, temperature, etc. must be maintained for months or even years to fulfill particular plant management or regulatory requirements.

Historical data is also frequently incorporated into trending packages to allow a graphical representation of data from a given period of time.

What types of Historical Data can be saved in the ControlWave Controller?

There are two types of historical data which can be saved within the ControlWave controller: **Archives** and **Audit Trail Logs**.

- **Archives** are snapshots of selected variables at a given moment in time. Archives are typically used for saving data which is destined for printed reports, such as flow variables, temperature variables, etc. Each archive record consists of a **timestamp** plus several columns of data values reflecting the state of process variables at the time of the timestamp, or in some cases, calculated values based on the state of process variables. Additional data for proper sequencing of the archive records is also stored. Typically, archives are generated at a pre-defined interval, however, on-demand archiving can also be configured. Archives are configured using both the Archive Configuration web page and the ARCHIVE function block.

- **Audit Trail Logs** are records of significant events occurring in the controller. There are two types of Audit Trail logs - - the alarm log and the event log. The **alarm log** maintains a record of any alarms generated in the ControlWave controller. The **event log** keeps a record of any changes to variables which have been designated for event monitoring. Other events which are logged include a System Date/Time change, recovery from a power failure, and 'note' events received from the human machine interface (HMI) software. Variables are designated for event monitoring by including them in an event list. Audit Trail logs are configured using both the Audit Configuration web page, and one or more AUDIT function block(s).

How is Audit Trail and Archive Data Retrieved from the ControlWave Controller?

OpenBSI Utilities, such as DataView, can collect Audit Trail and Archive data from the ControlWave unit and display it on the screen.

Alternatively, tools such as the OpenBSI Scheduler or OpenBSI Data Collector can retrieve the data and store it in historical data files, on a scheduled or demand basis. These files can then be automatically exported, using the OpenBSI Data File Conversion Utility, to CSV or ODBC-compatible file formats for use in OpenEnterprise, or in third-party software packages such as Microsoft® Excel® and Access® databases.

I/O Configurator

In order to reference **I/O points** on the process I/O boards of your controller, you need to configure them within your project.

Although it is *possible* to manually edit the “**IO_Configuration**” section of the project tree, we strongly recommend you use the I/O Configuration Wizard, as it will perform syntax checking, and is easier for most users.

The I/O Configuration Wizard is accessible from within ControlWave Designer by clicking: **View→IO Configurator**

When started, any existing I/O configuration data will be read and displayed in the I/O Configuration Wizard. The Configuration Wizard is a multi-page tool; [**>>Next>>**] and [**<<Back<<**] buttons are provided to allow you to move between the pages. A [**Settings**] push button allows the user to rename default variable names, if necessary. (See *Changing Default Variable Names*, later in this section.) NOTE: Page 1 allows the user to define multiple resources. Typically, only a single resource is used, so by default, page 2 will appear first since most users do not need to use Page 1.

Important

The IO Configuration Wizard will add a variable group to the Global_Variables worksheet called *IO_GLOBAL_VARIABLES*. Both the *IO_GLOBAL_VARIABLES* group in the Global_Variables worksheet and the IO_Configuration worksheet should never be manually edited by the user; these should only be modified through the IO Configuration Wizard.

CAUTION

If you intend to run multiple copies of ControlWave Designer simultaneously, **do not** attempt to run multiple copies of the I/O Configurator. If you do, you risk corrupting your I/O definitions.

Number of I/O Boards That May be Defined

Prior to OpenBSI 5.5 Service Pack 2, the number of I/O boards that could be defined within the I/O Configurator was 51. In OpenBSI 5.5 Service Pack 2 and newer, that limit has been increased to whatever number of boards can be defined in the 64K I/O memory space. In either case this limit includes both boards defined as local I/O as well as boards defined in a ControlWave I/O Expansion Rack or ControlWave Ethernet I/O unit.

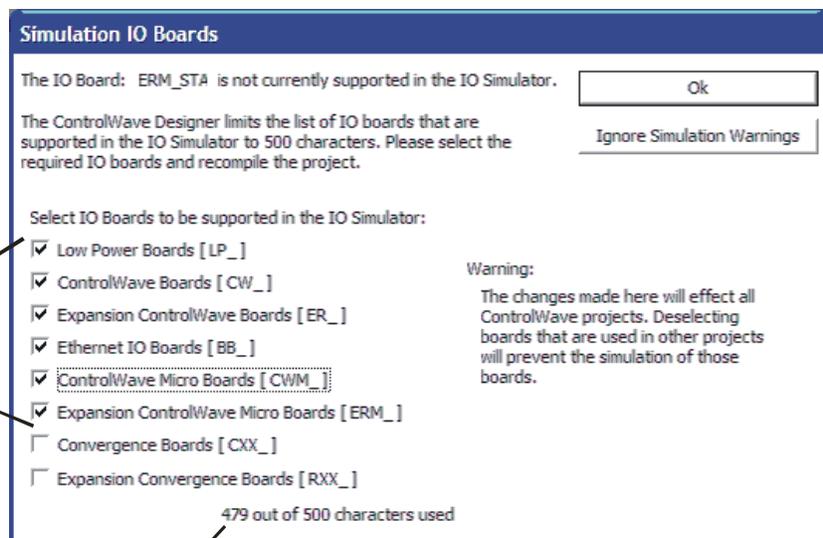
The larger of the input or output I/O map size of the board defines the overall board size. See the *I/O Mapping* section for details.

Effect of Number of Boards on the I/O Simulator

The I/O Simulator has a limit on the number of boards that may be accessible within a given simulation. This limit is dictated by an internal limit of 500 characters for the list used to describe the boards available during a given session within the I/O Simulator.

A default set of boards are normally included in the list available for simulation. If, after you've added boards in the I/O Configurator, you receive a warning message (see figure below) indicating that one or more of the boards you have selected is not supported within the I/O Simulator, you must de-select one or more of the default groups of boards that you are *not* using, to make additional room in the simulation list. Then select the group of boards that you *want* to use in the simulator. If groups for all the boards you need have now been selected, and you have de-selected enough of the unused board groups so that the total number of characters has fallen below 500, click **[Ok]** and you can proceed.

By default, these boards are included in the list of boards for the simulation.

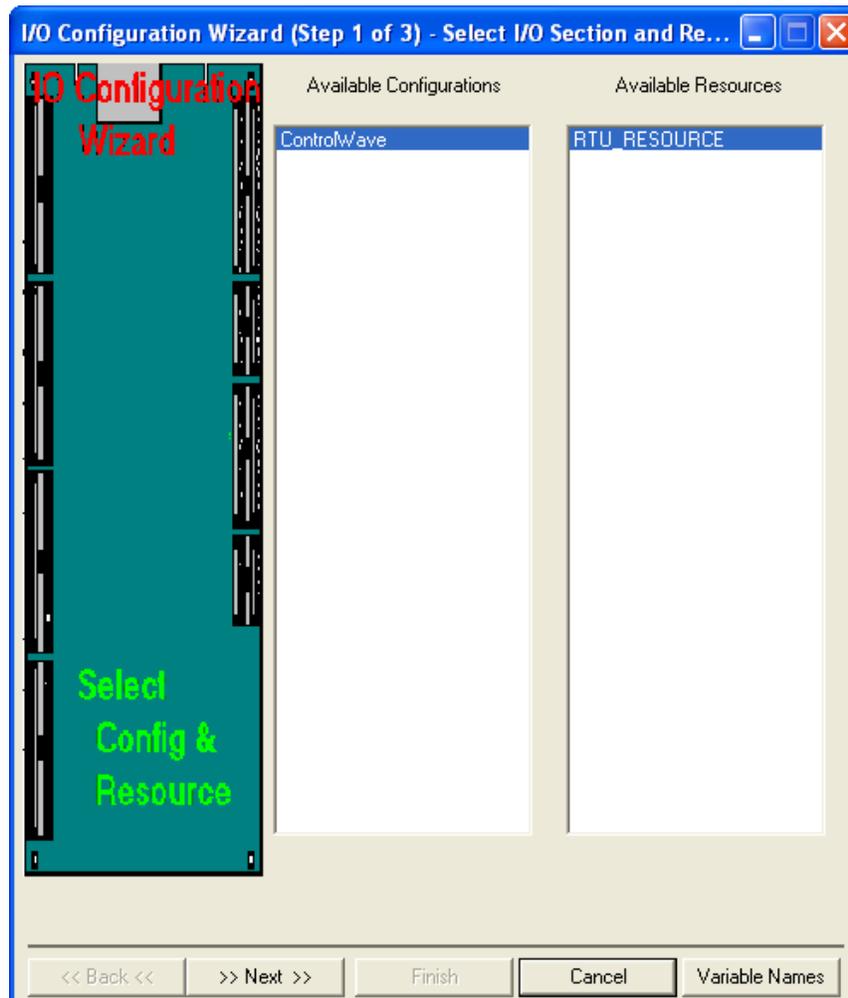


The maximum number of characters currently used. If this exceeds 500, you must un-check some boards to make room in the I/O Simulator list.

Note:

If you're not planning on using the I/O Simulator, you can click on **[Ignore Simulation Warnings]** and the I/O Configurator will allow you to proceed through this session, without checking for unsupported boards. If later, you decide you do want to use the I/O Simulator, and some boards do not appear, you must re-run the I/O Configurator, and de-select a sufficient number of board groups, and select the board groups you need, as described above.

I/O Configuration Wizard (Step 1 of 3): (Most users can skip to Step 2)



The first page of the I/O Configuration Wizard allows the user to select from the available I/O configurations and I/O resources. *NOTE: Because most projects use a single configuration and resource, this page is skipped when first starting the I/O Configuration Wizard. It is accessible, however, by clicking the [<<Back<<] button from the second page of the Wizard.*

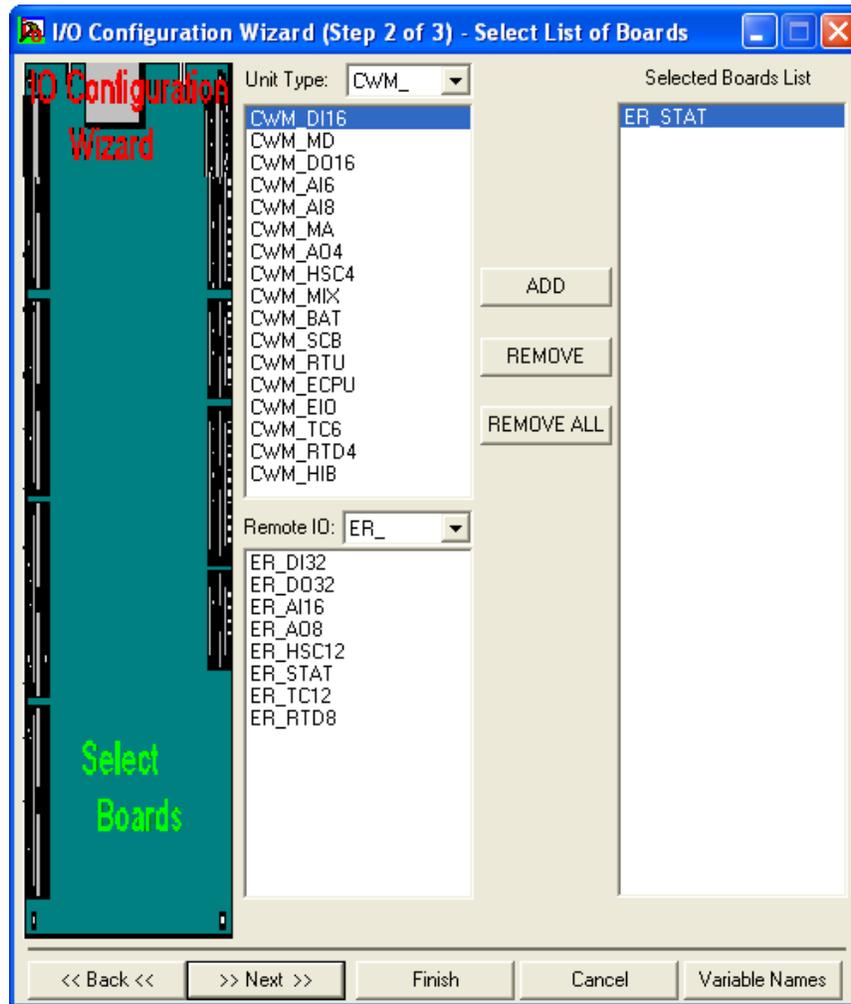
Available Configurations This lists all configurations in the current project. Select the I/O Configuration Section for which you are defining the I/O. *NOTE: Typically, projects use a single I/O configuration section.*

Available Resources This lists all resources for the selected I/O configuration. Choose the resource for which I/O is to be defined. **Note:** *Typically, projects use a single resource.*

Click [Next>] to proceed to the next step.

I/O Configuration Wizard (Step 2 of 3):

The second page of the I/O Configuration Wizard allows the user to identify which process I/O boards are actually installed in the ControlWave-series controller, as well as boards which are installed in separate devices such as I/O Expansion Racks, or Remote Ethernet I/O units.



Boards should be selected from the selection boxes in the ascending order of their slot number.

First, use the **“Unit Type”** list box to identify which type of ControlWave controller you are configuring, then select the desired boards, and click on **[ADD]**.

If this controller has associated I/O racks, or Remote Ethernet I/O units, choose those boards in the **“Remote IO”** selection box and click on **[ADD]**.

For more information on the various fields, see below:

Unit Type This field allows you to identify the type of ControlWave-series controller you are configuring, so that the proper board types can be

displayed for it. The types of controllers include:

CW_	ControlWave Process Automation Controller
CWM_	ControlWave MICRO Process Controller series
LP_	ControlWave Low Power (LP) Process Controller
CXX_	ControlWave CW_30 or CW_10 Controller
ERM_	Expansion Rack for ControlWave MICRO
RXX_	CW_35 Controller or CW_31 Remote I/O Rack

Once you select the type of controller, the boards which can be installed in that unit will be displayed as possible choices.

For ease of configuration, select the boards from the list in ascending order of their slot number in the ControlWave unit. Clicking once on the board abbreviation will cause a description of the board to be displayed at the bottom of the Wizard page. Double-clicking on the board abbreviation (or clicking *once* on the board and then clicking **[ADD]**) will add the board to the “Selected Boards List”. The table, on the next page, lists the various types of boards.

Remote IO

This lists boards used in ControlWave Remote Ethernet I/O units or ControlWave I/O Expansion racks.

Double-clicking on the board abbreviation (or clicking *once* on the board and then clicking **[ADD]**) will add the board to the “Selected Boards List”

Selected Boards List

This list allows the user to declare which boards reside in the ControlWave controller or its configured ControlWave Remote Ethernet I/O unit(s), or ControlWave I/O Expansion Racks. To remove a board from the “Selected Boards List” double-click on it, or click on it once, and then click **[REMOVE]**. To remove all boards click **[REMOVE ALL]**.

Click on **[Next]** to verify configuration information, adjust slot numbering, define zeros and spans for analog inputs, etc.

Tables of Board Types

ControlWave Process Automation Controller (CW)

Board Code	Board Description
CW_AI16	8 or 16 Input Pin Analog Board
CW_AO8	4 or 8 Output Pin Analog Board
CW_DO32	16 or 32 Output Pin Digital Board
CW_DI32	16 or 32 Input Pin Digital Board
CW_HSC12	6 or 12 Channel High Speed Counter/Universal Disc. Input Board
CW_RTD8	8 RTD Input Board
CW_TC12	12 Thermocouple Input Board

ControlWave MICRO Process Controller (CWM) – series

Board Code	Board Description
CWM_AI6	6 Input Pin Analog Board
CWM_AI8	8 Analog Input Board
CWM_AO4	4 Analog Output Board
CWM_BAT	Battery (Voltage) Monitor
CWM_DI16	16 Input Pin Digital Board
CWM_DO16	16 Output Pin Digital Board
CWM_ECPCU	System Controller Board
CWM_EIO	Mixed I/O Board (various configuration options)
CWM_HIB	HART Interface Board (HIB)
CWM_HSC4	4 Channel High Speed Counter
CWM_MA	Mixed Analog Board (6 analog inputs, 2 analog outputs)
CWM_MD	Mixed Digital Board (12 digital inputs, 4 digital outputs)
CWM_MIX	Mixed I/O Board
CWM_RTD4	4 Resistance Temperature Device (RTD) Input Board
CWM_RTU	Mixed I/O & System Controller Board
CWM_SCB	System Controller Board
CWM_TC6	6 Thermocouple Input Board

Some ControlWave MICRO boards are used in multiple platforms. See the table, below for details:

	CW MICRO	CW EFM	CW XFC	CW GFC CL	CW GFC Plus	CW GFC	CW Corrector	CW Express	CW EPAC	CW GFC-IS
CWM_AI6	■	■								
CWM_AI8	■	■								
CWM_AO4	■	■								
CWM_BAT*	■	■								
CWM_DI16	■	■								

	CW MICRO	CW EFM	CW XFC	CW GFC CL	CW GFC Plus	CW GFC	CW Corrector	CW Express	CW EPAC	CW GFC-IS
CWM_DO16	■	■								
CWM_ECPU						■	■	■	■	■
CWM_EIO						■	■	■	■	■
CWM_HIB	■	■								
CWM_HSC4	■	■								
CWM_MA	■	■								
CWM_MD	■	■								
CWM_MIX	■	■								
CWM_RTD4	■	■								
CWM_RTU			■	■	■					
CWM_SCB **	■	■								
CWM_TC6	■	■								

* Does not support wet end.

** Supports wet end.

ControlWave 10/30 Controllers (CW_10, CW_30)

Board Code	Board Description
CXX_AI8	4 or 8 Analog Input Board
CXX_AO4	2 or 4 Analog Output Board
CXX_DI16	8 or 16 Digital Input Board
CXX_DO16	8 or 16 Digital Output Board
CXX_HSC8	4 or 8 Channel High Speed Counter Board
CXX_LL4	4 Low Level Analog Input Board

ControlWave 35/31 Controller and I/O Rack (CW_35, CW_31)

Board Code	Board Description
RXX_AI8	4 or 8 Analog Input Board
RXX_AO4	2 or 4 Analog Output Board
RXX_DI16	8 or 16 Digital Input Board
RXX_DO16	8 or 16 Digital Output Board
RXX_HSC8	4 or 8 Channel High Speed Counter Board
RXX_LL4	4 Low Level Analog Input Board
RXX_STAT	External Rack Status Board

ControlWave Low-Power Controller (LP)

Board Code	Board Description
LP_AI8	8 Input Pin Analog Board (Slot 0 – fixed)
LP_AO4	4 Output Pin Analog Board
LP_BAT	Battery (Voltage) Monitor (Slot 0 – fixed)

Board Code	Board Description
LP_DI16	16 Input Pin Digital Board (Slot 0 – fixed)
LP_DO8	8 Output Pin Digital Board (Slot 0 – fixed)
LP_HSC4	4 Channel High Speed Counter (Slot 0 – fixed)

ControlWave I/O Expansion Rack (ER)

Board Code	Board Description
ER_AI16	16 Analog Input Board
ER_AO8	8 Analog Output Board
ER_DI32	32 Digital Input Board
ER_DO32	32 Digital Output Board
ER_HSC12	12 Channel High Speed Counter Board
ER_RTD8	8 Resistance Temperature Device (RTD) Input Board
ER_STAT	I/O Expansion Rack Statistics Board (Virtual board) NOT A PHYSICAL HARDWARE BOARD
ER_TC12	12 Thermocouple Input Board

Expansion Rack ControlWave MICRO

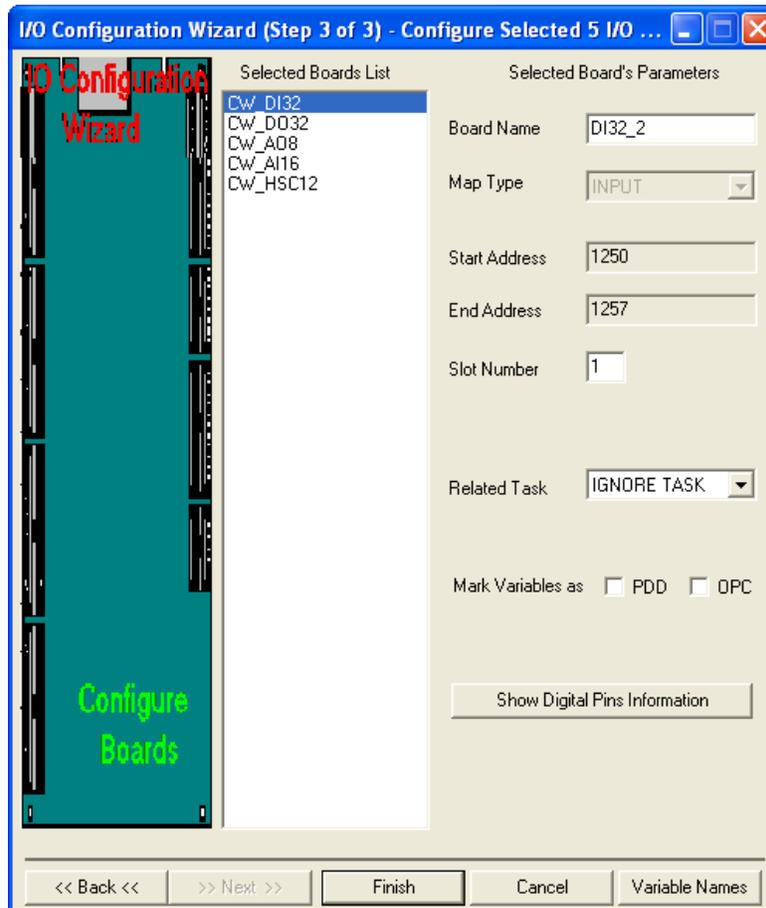
Board Code	Board Description
ERM_AI6	6 Analog Input Board
ERM_AI8	8 Analog Input Board
ERM_AO4	4 Analog Output Board
ERM_DI16	16 Digital Input Board
ERM_DO16	16 Digital Output Board
ERM_HSC4	4 High Speed Counter Board
ERM_MA	6 Analog Input and 2 Analog Output Board
ERM_MD	12 Digital Input and 4 Digital Output Board
ERM_MIX	Mixed I/O Board
ERM_RTD4	4 Resistance Temperature Device (RTD) Input Board
ERM_STAT	External Rack Status Board
ERM_TC6	6 Thermocouple Input Board

ControlWave Ethernet Remote I/O (BB)

Board Code	Board Description
BB_16AI	RIO- 16AI2 (16 Remote Analog Input)
BB_8AI4AO	RIO- 8AI2, 4AO2 (8 Remote Analog Input <i>and</i> 4 Remote Analog Output)
BB_8DI8AI	RIO- 8DI2 – 8AI2 (8 Remote Digital Input <i>and</i> 8 Remote Analog Input)
BB_8DI8DO	RIO- 8DI2 – 8DO2 (8 Remote Digital Input <i>and</i> 8 Remote Digital Output)
BB_16DI	RIO- 16DI2 (16 Remote Digital Input)
BB_16DO	RIO- 16DO2 (16 Remote Digital Output)
BB_8INS	RIO- 8INS (Instrumentation Board)
BB_8HSC	RIO - 8HSC (8 channel high speed counter)
BB_4RTDI	RIO 4RTD - 4 Digital Input Board
IPMB_INP	RIO Open Modbus Input
IPMB_OUT	RIO Open Modbus Output

I/O Configuration Wizard (Step 3 of 3):

The third page of the I/O Configuration Wizard displays configuration details for each board. To see the details, click on the board abbreviation, and the configuration details will be displayed on the right hand side of the page.



Selected Boards List Displays all boards selected on the previous page. Click on a particular board abbreviation to display configuration details for the board.

Board Name A name for the board can be specified here. This name will be used when configuring pins for the board.

Map Type (Information only field) Depending upon the type of board, separate memory areas (called maps) are reserved for either inputs or outputs. Some boards have both an input map *and* an output map. For example, a digital output board has outputs (DOs) in its output map, but it may also have inputs which indicate board status conditions and errors.

Note: *If you have an older ControlWave project in which you changed the map type from the default choice, this may cause errors to be generated when the project is rebuilt. If this occurs, you should delete the board definition and re-define the board.*

For more detailed information on the input and output maps for various boards, see the 'I/O Mapping' section of this manual.

Start... End Address	Displays the range of memory addresses used by the board.
Slot Number	Displays either the physical I/O slot in the ControlWave controller which holds the board, or if this is a Mixed I/O Board (MIOB) it displays a board selection number. For ControlWave and ControlWave Micro, I/O slot numbers are positive integers, e.g. 1, 2, 3, etc. For the ControlWaveLP, the slot number is 0 for all boards except for the AO; for the AO the slot can be 8 to 13. <i>NOTE: I/O Slot number is NOT the same as the chassis slot number. Chassis slots which hold the power supply and CPU boards are not considered to be I/O slots, so the first I/O slot is typically the third chassis slot.</i>
IP Address	ControlWave Remote Ethernet I/O boards are identified by their Internet Protocol (IP) address, instead of the I/O slot number. The same is true for boards residing in a ControlWave I/O Expansion Rack.
Related Task	Shows the name of the task which uses this board. In some cases, for example, when using Ethernet I/O, or analog boards in an RTU 3340, it is important to associate a board with the task which uses the board. When a board is associated with a task, that board will be read / written to, at the rate cycle associated with the task, thereby ensuring up-to-date information for calculations performed in the task. When no task is associated with the board, board execution is associated with the default task, which runs at a lower priority, and therefore may not provide sufficient up-to-date I/O information when it is required by a task.
Mark Variables as PDD OPC	This determines how values of the I/O variables associated with this board will be made available to other software programs. Checking "PDD" allows the controller to reference variables by name, which is necessary if you intend to access a variable by external software which requires 'read-by-name' access, such as DataView, or one of the other OpenBSI Utilities. Checking "OPC" adds this variable to a collection list used by the OPC Server or by the OpenBSI Signal Extractor. This is necessary when data is to be extracted, and sent to a database.

When edits have been made to this page, click on the **[Show xxx Information]** button. The name on this button, and the pin configuration details, vary depending on the type of board being configured. See the pages that follow for the standard board types.

Analog Boards

The screenshot shows the 'Configure List of Available Analog Pins' dialog for an Analog Input Board (CWM_AI8). The dialog has a title bar 'Configure List of Available Analog Pins'. It is divided into two main sections: 'List of Available Pins' and 'Pin Properties'.
 - **List of Available Pins:** A list of pins from PIN : 1 to PIN : 8. PIN : 1 is selected and highlighted in blue. Below the list is a 'Pin Used' label.
 - **Pin Name:** A text field containing 'AI8_5_1'.
 - **Pin Properties:** A section containing several fields and checkboxes:
 - 'Zero' field: 0.000000
 - 'Span' field: 100.000000
 - 'Add Overrange Status' checkbox: unchecked
 - 'Range Type' dropdown: set to 'VOLTS'
 - 'Bottom Range' field: 1.000000
 - 'Top Range' field: 5.000000
 - **Right Side:** A 'Done' button and four checkboxes:
 - 'Add Board Status' (unchecked)
 - 'Calibration Error' (unchecked)
 - 'Board Time Out' (unchecked)
 - 'Mark All Pins Used' (unchecked)

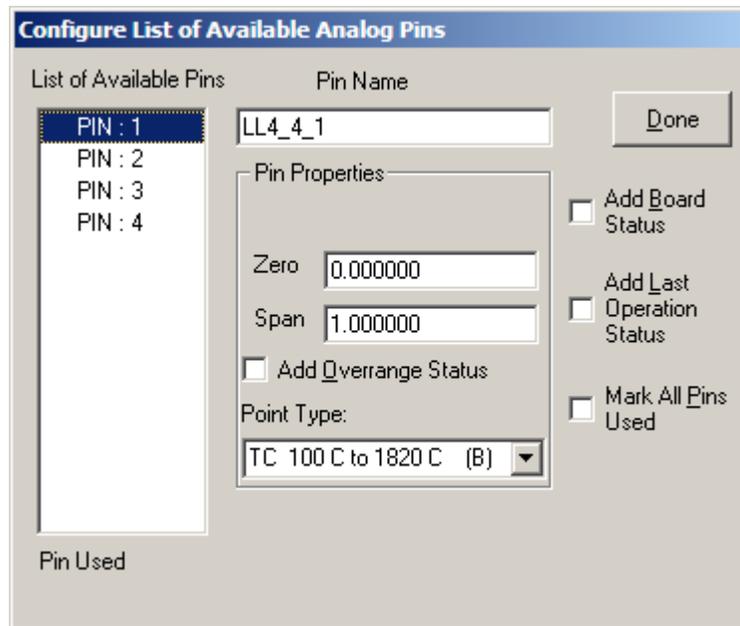
Analog Input Board Page (CWM_AI8 board)

The screenshot shows the 'Configure List of Available Analog Pins' dialog for an Analog Output Board (CW_AO8). The dialog has a title bar 'Configure List of Available Analog Pins'. It is divided into two main sections: 'List of Available Pins' and 'Pin Properties'.
 - **List of Available Pins:** A list of pins from PIN : 1 to PIN : 8. PIN : 1 is selected and highlighted in blue. Below the list is a 'Pin Used' label.
 - **Pin Name:** A text field containing 'AO8_1_1'.
 - **Pin Properties:** A section containing several fields and checkboxes:
 - 'Value' field: 0.000000
 - 'Zero' field: 0.000000
 - 'Span' field: 100.000000
 - 'Add Overrange Status' checkbox: unchecked
 - 'Set Actual Output Value' checkbox: unchecked
 - 'Configure Hold Values' checkbox: checked
 - 'Update Default Value' checkbox: unchecked
 - 'Hold Last Output' checkbox: unchecked
 - 'User Configured Output' field: 0.000000
 - **Right Side:** A 'Done' button and three checkboxes:
 - 'Add Board Status' (unchecked)
 - 'Add Last Operation Status' (unchecked)
 - 'Mark All Pins Used' (unchecked)

(some of these fields do NOT appear for other models)

Analog Output Board Page (CW_AO8 board)

(some of these fields do NOT appear for certain models)



Low Level Analog Input Board

List of Available Pins

Displays a list of the individual pins (I/O points) on this process I/O board. If the pin is displayed in RED, that pin is active. If the pin is left grayed out, that pin is considered unused.

Pin Name

Defines a name identifying this pin. IMPORTANT: This name is used as a variable name to reference the I/O pin in your POU.

Value

Defines the initial value for this I/O pin, in floating point format. NOTE: This is not available for analog input pins.

Zero

Defines the lowest value of the range for this I/O pin. Used to scale the input/output value.

Span

Span is added to the ZERO value to define the highest value of the range for this I/O pin. Used to scale the input/output value.

Add Over Range Status

When selected, will cause a variable to be created to store the value of the overrange status bit. Over range conditions occur when an attempt is made to drive the variable associated with this pin outside the range defined by the zero and span. When this occurs, the over range status bit will be set to TRUE.

Range Type

Some boards allow you to specify whether the board input is in current or voltage. Choose 'VOLTS' or 'AMPS'. NOTE: For example, if 4 to 20 milliamps of current drive the board, you would choose 'AMPS', then enter 0.004 for the "Bottom Range" value, and 0.020 for the "Top Range" value.

Bottom Range	The lowest usable value for VOLTS or AMPS for this board input. For example, if the board input can range from 1 to 5 VOLTS, the "Bottom Range" would be set to 1.0. If this board input can range from 4 to 20 milliamps, "Bottom Range" would be set to 0.004. Other ranges are possible as well.
Top Range	The highest usable value for VOLTS or AMPS for this board input. For example, if the board input can range from 1 to 5 VOLTS, the "Top Range" would be set to 5.0. If this board input can range from 4 to 20 milliamps, "Top Range" would be set to 0.020. Other ranges are possible as well.
Set Actual Output Value	When selected, this will cause a variable to be created which <i>displays</i> the actual value which was written to the output pin.
Add Board Status	When selected, will cause a variable to be created to store board status information.
Add Last Operation Status	When selected, will cause a variable to be created to store the status of the last conversion operation information.
Calibration Error	This is only present for certain ControlWave MICRO boards. When checked, will cause a variable to be created to store error information. This variable will be set to TRUE whenever there is bad calibration data in the EEPROM.
Board Time Out	This is only present for certain ControlWave MICRO boards. When checked, will cause a variable to be created to store information about board time out errors. Board time outs occur if there is a problem with conversion operations.
Mark All Pins Used	When checked, will activate all pins on this I/O board. They will all appear in RED.
Configure Hold Values	When checked, enables other fields on the page for configuring a hold value for this pin. A hold value is the value used by the I/O card if it detects a watchdog of the ControlWave CPU. The I/O board maintains this value at the pin until the unit is restarted.
Update Default Value	When checked, allows the "User Configured Output" hold value to be changed on-line; otherwise the hold value can only be set in the I/O Configurator.
Hold Last Output	When checked, specifies that during a watchdog failure, the hold value for this pin will be whatever value was on the pin when the failure occurred. NOTE: "Hold Last Output" and "User Configured Output" are mutually exclusive. Either one may be configured for a particular pin, but NOT both.
User Configured Output	When checked, allows the user to enter a value for this pin which will be used as the hold value in the event there is a watchdog

failure of the ControlWave. NOTE: "Hold Last Output" and "User Configured Output" are mutually exclusive. Either one may be configured for a particular pin, but NOT both.

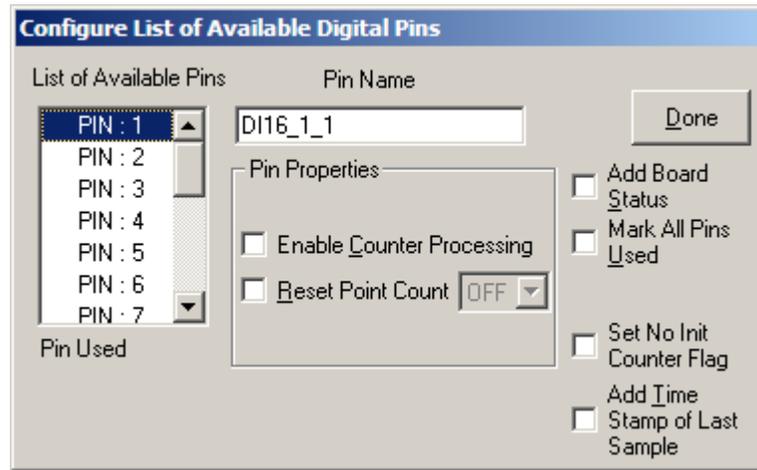
Point Type

Specifies the type of low-level analog input/thermocouple. See the table, below, for a list of supported temperature/voltage ranges for inputs to the board.

Point Type	Range
Thermocouple Type B	100°C to 1820°C
Thermocouple Type E	-270°C to 1000°C
Thermocouple Type J	-210°C to 1200°C
Thermocouple Type K	-270°C to 1370°C
Thermocouple Type R	-50°C to 1720°C
Thermocouple Type S	-50°C to 1760°C
Thermocouple Type T	-270°C to 400°C
Resistance Temperature Device (RTD)	-220°C to 850°C
Voltage	-10 mV to 10 mV

When all pins have been configured, click on **[Done]**. You can then proceed to select and configure pins for *another* board.

Digital Boards



Digital Input Board Page
(Not all fields are present for all board types)



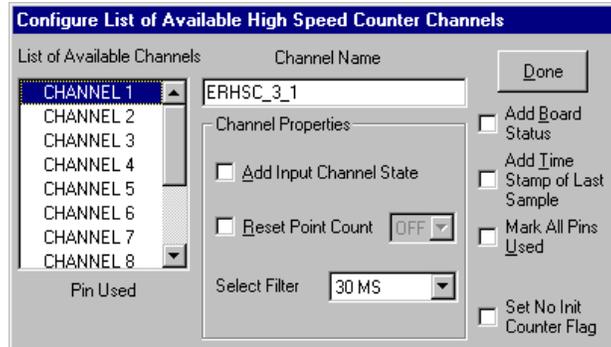
Digital Output Board Page

List of Available Pins	Displays a list of the individual pins (I/O points) on this process I/O board. If the pin is displayed in RED, that pin is active. If the pin is left grayed out, that pin is considered unused.
Pin Name	Is a name identifying this pin. This name is used as a variable name to reference the I/O pin in your POU.
Set Pin Status	Sets the initial value for this digital output (DO). NOTE: This option is not available for digital inputs.
Enable Counter Processing	Turns on or off the counters associated with the digital input (DI) process I/O board. Counters are used in certain applications. For example, if a mixed I/O board is used with a ControlWaveLP, a digital input (DI) can be used as a low speed counter (30 millisecond filter). Enabling counter processing in such a case will allow interrupt processing to occur for that DI.

Add Board Status	When selected, will cause a variable to be created to store board status information.
Mark All Pins Used	When checked, will activate all pins on this I/O board. They will all appear in RED.
Turn off Leds	This option is only available on certain ControlWave MICRO boards. When checked, it will create a variable which allows you to turn OFF the I/O board's diagnostic LEDs to save on power. LEDs are turned OFF when the variable is set ON. <i>NOTE: For this to work, the LED enable jumper on the board must be in position 2-3; otherwise, the software cannot disable the LEDs, only a hardware jumper can. See manual CI-ControlWaveMICRO for details.</i>
Reset Point Count	When set to ON, allows the number of counts to be reset. This occurs automatically whenever the board is restarted.
Set No Init Counter Flag	When checked, counters on the board will NOT be initialized to zero on a warm start of the unit.
Add Time Stamp of Last Sample	When selected, will cause a variable to be created to store the timestamp of the last sample collected by this I/O board.

When all pins have been configured, click on **[Done]**. You can then proceed to select and configure pins for *another* board.

High Speed Counter (HSC) Boards



High Speed Counter Page

(Not all fields are present for all board types / platforms)

List of Available Channels	Displays a list of the individual channels (counter I/O points) on this process I/O board. If the channel is displayed in RED, that channel is active. If the channel is left grayed out, that channel is considered unused.
Channel Name	Is a name identifying this channel. This name is used as a variable name to reference the channel in your POU.
Add Input Channel State	When selected, displays the TRUE/FALSE value of the channel.
Reset Point Count	When selected allows the number of counts to be reset. Choose either ON or OFF for the initial value on startup. A reset occurs when you choose ON; software then turns this OFF. NOTE: Reset occurs automatically whenever the board is restarted.
Select Filter	Specifies how the board will operate for this channel: <ul style="list-style-type: none"> 'None' Defaults to 30 millisecond filtering. '30 ms' Turns on 30 millisecond filter. Typically used for push-button debouncing. '1 ms' Turns on 1 millisecond filter. Used for low speed counter applications. 'HSC Channel' High Speed Counter. 10 KHz filter. (Default for CWM_RTU board). Requires 04.90 or newer firmware.
Add Board Status	When selected, will cause a variable to be created to store board status information.
Add Time Stamp of Last	When selected, will cause a variable to be created to store the timestamp of the last sample collected by this I/O board.

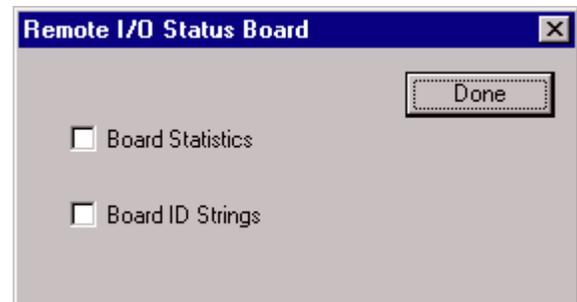
Sample

- Mark All Pins Used** When checked, will activate all channels on this I/O board. They will all appear in RED.
- Set No Init Counter Flag** When checked, counters on the board will NOT be initialized to zero on a warm start of the unit. Requires 04.41 or newer firmware.

Turn off Leds (Not Shown) This option is only available on certain ControlWave MICRO boards. When checked, it will create a variable which allows you to turn OFF the I/O board's diagnostic LEDs to save on power. LEDs are turned OFF when the variable is set ON. *NOTE: For this to work, the LED enable jumper on the board must be in position 2-3; otherwise, the software cannot disable the LEDs, only a hardware jumper can. See manual CI-ControlWaveMICRO for details.*

Remote I/O Status Board

The Remote I/O Status Board is a 'virtual' board, i.e. there is no actual physical board. By including it within your ControlWave project, global variables will be created to store communication statistics information, and board ID strings for the ControlWave I/O Expansion Rack, or other remote I/O devices.

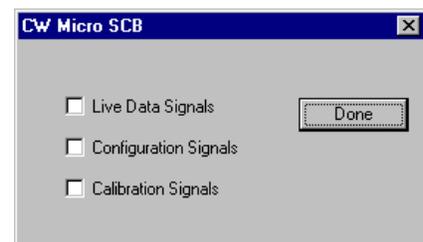
**Note:**

The size of RIO STAT boards has increased. This can cause an overlap with the memory maps of other I/O boards. If you have an RIO STAT board in your project, please remove it, and then add it back into the project, to allow memory maps to be adjusted properly.

For more information about these RIO status variables, and the software configuration for the ControlWave I/O Expansion Rack, please see the *ControlWave I/O Expansion Rack Quick Setup Guide* (document# D5122).

System Controller Board

The System Controller Board is used with ControlWave MICRO units equipped with a transmitter, such as the Electronic Flow Meter (EFM) version of the ControlWave MICRO.



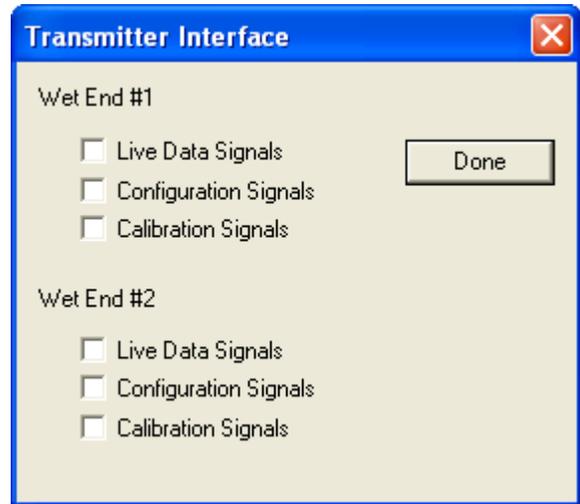
Live Data Signals	When selected, creates variables for storing live data, e.g. pressure readings, from the transmitter.
Configuration Signals	When selected, creates variables for storing configuration information.
Calibration Signals	When selected, creates variables for storing calibration information.

CWM_RTU Board

In addition to analog and digital pins, certain RTUs with the CWM_RTU board (GFC, XFC) may include a built-in internal transmitter with sensor (wet end). Some special versions of the XFC can include two wet ends.

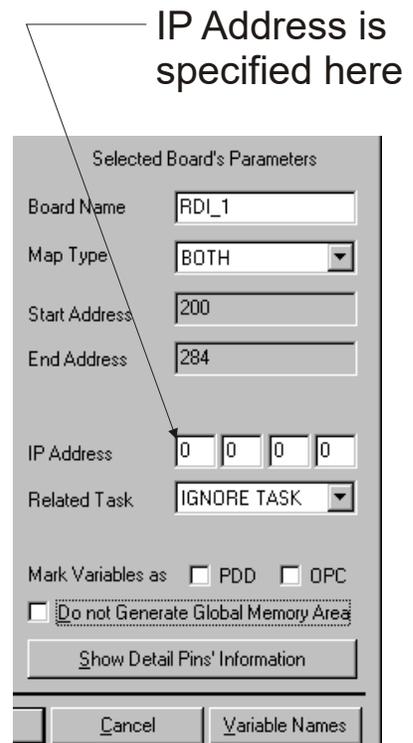
The Transmitter Interface dialog box for the CWM_RTU allows variables to be mapped for both of the wet ends.

These choices are similar to the System Controller Board.



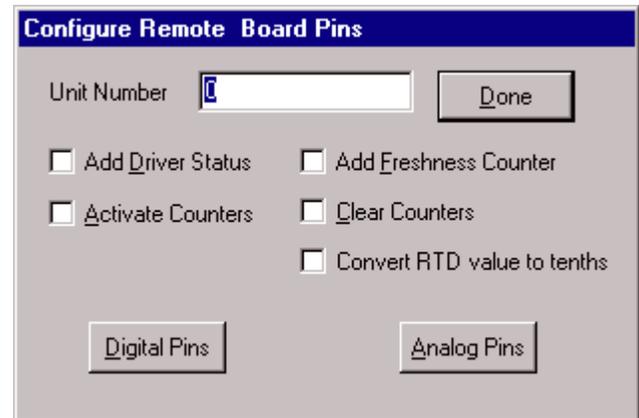
Notes About Ethernet I/O Boards

Unlike process I/O boards which are physically installed in the ControlWave controller, ControlWave Remote Ethernet I/O boards are in a separate location, and communicate to the ControlWave unit using TCP/IP. (The IP address for the Ethernet I/O board is configured from the third page of the I/O Configuration Wizard.)



Certain parameters must be specified for the Ethernet I/O units which hold the boards. Once this is done, however, the configuration of the individual board pins is identical to that described earlier.

Note: The dialog box shown at right includes all possible fields for the ControlWave Remote Ethernet I/O, however, not all of these fields are visible in all cases.



Unit Number	Specifies the Modbus unit address number associated with this ControlWave Remote Ethernet I/O unit.
Add Driver Status	When selected, will cause a variable to be created to store I/O driver status information.
Activate Counters	Creates / disables a variable which allows the user to control the starting / stopping of the counters in the ControlWave Remote Ethernet I/O board. These counters are used with digital inputs (DI).
Add Freshness Counter	When selected, will cause a variable to be created to store a 'freshness' counter value. The freshness counter represents the number of program executions since new data has been collected through this Ethernet I/O board. A value of 0, indicates the data is as new (fresh) as possible.
Clear Counters	Sets all counter values associated with this board to 0.
Convert RTD value to tenths	(For RIO 4RTD - 4 Digital Input Board ONLY) - When checked, causes values from the Resistance Temperature Device board to be divided by 10, thereby providing greater precision.
[Show Pins], [Analog Pins], [Digital Pins]	When clicked on, calls up a dialog box for configuring the individual pins for the board.
[Done]	Click here when configuration for this board is complete.

Additional Configuration For ControlWave Remote Ethernet I/O

Besides the I/O configuration within ControlWave Designer, additional configuration for Ethernet I/O hardware must be performed using the Remote I/O Toolkit software (*not to be*

confused with what used to be known as the OpenBSI Technician Toolkit). Documentation on the Remote I/O Toolkit software is provided in the form of on-line help screens.

The Remote I/O Toolkit software is included as an installation option on the OpenBSI CD ROM.

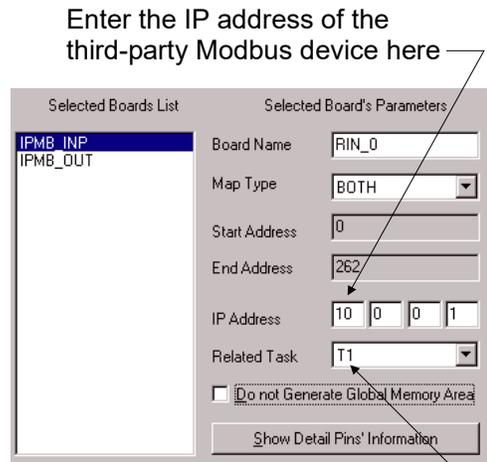
- To use counters (DI) you must enable counters in the Remote I/O Toolkit software.
- The IP address entered for a ControlWave Remote Ethernet I/O board in the ControlWave Designer I/O Configuration Wizard must MATCH the IP address entered in the Remote I/O Toolkit.
- If you are using high speed counters, 32 bit counters must be enabled within Remote I/O Toolkit.
- For analog inputs/outputs (AI, AO) you must NOT change the default scaling within Remote I/O Toolkit. Changes should only be made within the ControlWave Designer I/O Configuration Wizard.
- Be aware that if you are using counters (Digital Input or High Speed Counter), restarting of the ControlWave Remote Ethernet I/O will cause a large jump in counts.
- If you intend to use TPO (Time Proportioned Outputs) for any point, you must enable TPO for those points.
- If you check the 'Turn OFF outputs on communications loss' option' in the Remote I/O Toolkit, outputs will be set to 0 if the 'Com Timeout' value expires without any communication from the ControlWave. The default value for 'Com Timeout' in the Remote I/O Toolkit is 5 seconds. The rate at which the ControlWave communicates with the Ethernet I/O is determined by the ControlWave task associated with the board. If this is an 'output-only' board, however, and the output(s) coming from the ControlWave *have not changed*, the ControlWave will not attempt to communicate with the Ethernet I/O more frequently than once every 15 seconds. (Prior to ControlWave firmware 04.60, if outputs had not changed, the ControlWave would not attempt to communicate with the Ethernet I/O more frequently than once every 60 seconds.) To prevent outputs from being zeroed out due to a delay in communication from the ControlWave because outputs have not changed, you must increase the 'Com Timeout' value to *greater than* 15 seconds. The 'Com Timeout' value has a maximum of 25 seconds.

RIO Open Modbus Boards

These RIO Open Modbus board types are provided to allow the ControlWave to communicate with various third-party Modbus devices.

Before using these board types, you must be familiar with certain characteristics of the third-party device. In particular, you will need to know the following:

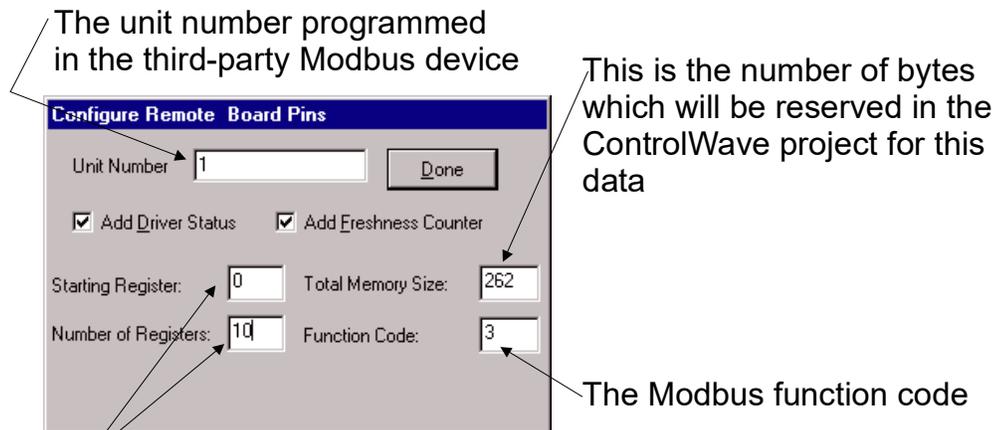
- The IP address of the third-party Modbus device. This is entered in the I/O Configurator as shown in the figure, below.



It is recommended that you associate the board with some executing task. Data collection from the Modbus device will occur at the rate specified for the task.

- The Modbus Unit Number as programmed in the third-party Modbus device.
- The register numbers in the third-party Modbus device which you will be 'reading from / writing to'.
- The Modbus function code(s) you will be using to 'read from / write to' registers in the third-party Modbus device. This also affects your choices of data types for the variables in ControlWave Designer which will be used to hold the Modbus register data.

These parameters are entered in the Configure Remote Board Pins dialog box, which is accessible from the **[Show Detail Pins' Information]** button.



These numbers define which registers will be 'read-from / written to' in the Modbus device. Here, we are requesting data from 10 registers (numbered 0 through 9).

Unit Number	Specifies the Modbus unit number as programmed in the third-party Modbus device. NOTE: If you will be requesting data from non-contiguous regions of memory in the same Modbus device, and you need to use the same function code to collect the data, you will need to define multiple Open Modbus boards, and give them different unit numbers, even though they all refer to the same Modbus device. The first board definition should use the actual Modbus unit number in the device; the unit numbers for all subsequent boards must be chosen by adding a multiple of 256 to the actual unit number. If, say, the Modbus unit number programmed in the Modbus device is 13, and you need to define three different boards to get data from three different memory regions in the device, you should use Modbus unit numbers of 7, 269, and 525. All refer to the same device.						
Add Driver Status	When selected, will cause a variable to be created to store I/O driver status information.						
Add Freshness Counter	When selected, will cause a variable to be created to store a 'freshness' counter value. The freshness counter represents the number of program executions since new data has been collected through this Open Modbus board. A value of 0, indicates the data is as new (fresh) as possible.						
Total Memory Size	This is the total number of bytes which will be reserved inside the ControlWave project for data 'read from / written to' the third-party Modbus device. The default is the maximum memory available for the longest possible request.						
Starting Register	The first register in the third-party Modbus device which you will be 'reading from / writing to'. IMPORTANT NOTE: Both the Open Modbus Input Board and Open Modbus Output Board send requests for the exact register number you specify. Some third-party Modbus devices, however, number their registers differently, for example, starting register numbers at the number 1, instead of the number 0. As a consequence, you may need to request one less than the register number you want to get the correct register. Consult the literature accompanying the device to verify the register numbering.						
Number of Registers	The total number of registers to be read/written.						
Function Code	The Modbus function code. Only the Modbus function codes listed, below, can be used through these boards: <table border="0" style="margin-left: 40px;"> <tr> <td style="padding-right: 20px;">1</td> <td>Read Multiple Coil Outputs</td> </tr> <tr> <td>2</td> <td>Read Multiple Coil Inputs</td> </tr> <tr> <td>3</td> <td>Read Multiple Analog Outputs</td> </tr> </table>	1	Read Multiple Coil Outputs	2	Read Multiple Coil Inputs	3	Read Multiple Analog Outputs
1	Read Multiple Coil Outputs						
2	Read Multiple Coil Inputs						
3	Read Multiple Analog Outputs						

4	Read Multiple Analog Inputs
15 (Fhex)	Write Multiple Coil Outputs
16 (10hex)	Write Multiple Analog Outputs

[Done] Click here when configuration for this board is complete.

Besides configuring the board, itself, you must explicitly declare located variables in one of your ControlWave worksheets that will hold the data 'read from / written to' the Modbus device.

Located variables would typically be entered either in your 'RTU_RESOURCEV' worksheet, or in some other worksheet you create, and would take the format shown below:

<i>variable_name</i>	<i>Datatype</i>	<i>usage</i>	<i>description</i>	<i>%location_prefixsize_prefixaddress</i>
----------------------	-----------------	--------------	--------------------	---

variable_name is the variable name.

datatype is one of the IEC 61131-3 data types, e.g. BOOL, INT, etc.

usage specifies the scope of how the variable is used, e.g. VAR, VAR_GLOBAL, etc.

description is an optional description.

location_prefix describes where this data will be located. It is one of the following letters:

I for physical inputs (input map)

Q for physical outputs (output map)

size_prefix specifies the amount of space needed for the variable. It is one of the following letters. (NOTE: If no *size_prefix* is included, single bit size is assumed.)

X single bit size (BOOL only)

B byte size (8 bits)

W word size (16 bits)

D double word size (32 bits)

address is the memory address reserved this variable plus the appropriate offset. Inputs and outputs start at an offset of 4 in the memory map of the board. See '*I/O Mapping*' for more details on offsets into the I/O map for the Open Modbus boards.

As an example, suppose we have defined both an Open Modbus Input board that will be reading a register in the Modbus device to obtain a flow temperature. This temperature is

Name	Type	Usage	Description	Address
[-] NewGroup				
FLOW_TEMP	INT	VAR_GLOBAL		%IW4

stored in the Modbus device as a 16 bit (word), and will be stored in an integer variable in the ControlWave, named FLOW_TEMP. The variable must be declared as follows:

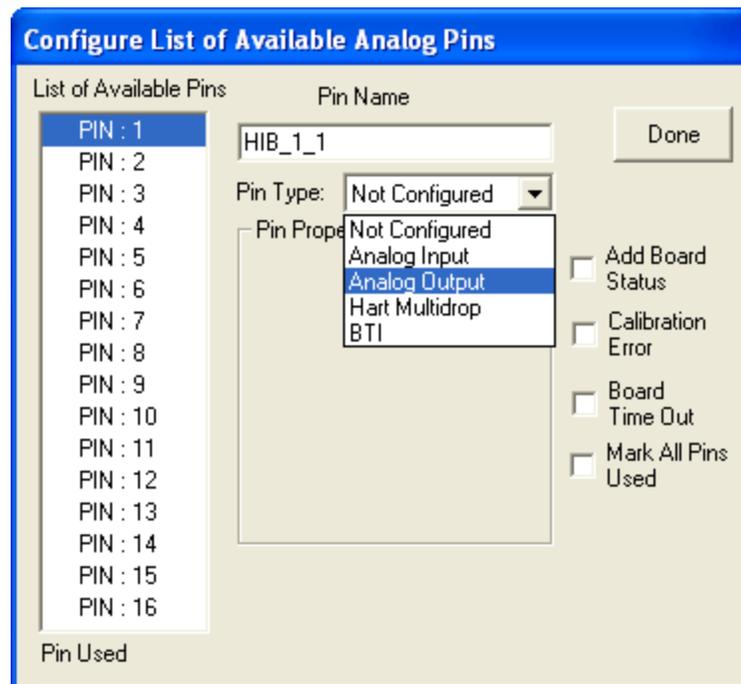
The choice of 4 for the address was determined by examining the I/O global variables worksheet to determine the address of the IPMB_INP (Open Modbus Input board). In this case, the address was 0, and then putting in the correct offset, based on the I/O Mapping information in the 'I/O Mapping' section. In this case the offset was 4.

HART Interface Board (CWM_HIB)

The CWM_HIB board allows your ControlWave MICRO to communicate with HART® devices using the Highway Addressable Remote Transducer (HART) protocol, or with Bristol 3508/3808 transmitters using the BSAP protocol.

If using a HART device, you must configure a HART function block. See the ControlWave Designer online help for details.

If using a Bristol 3508 or 3808 transmitter, you must configure an XMTR or LBTI function block. See the ControlWave Designer online help for details.



You must associate the HIB board with a task, using the **Related Task** field, in order to have control of its rate of execution.



It must be executed fast enough to accommodate your data update requirements, depending upon whether or not you are accessing the current loop directly. For example, if you want to access the primary variable at a faster rate (100 msec) through the 4 to 20 mA current loop, you must associate the HIB board with a task that executes at least once every 100 msec.

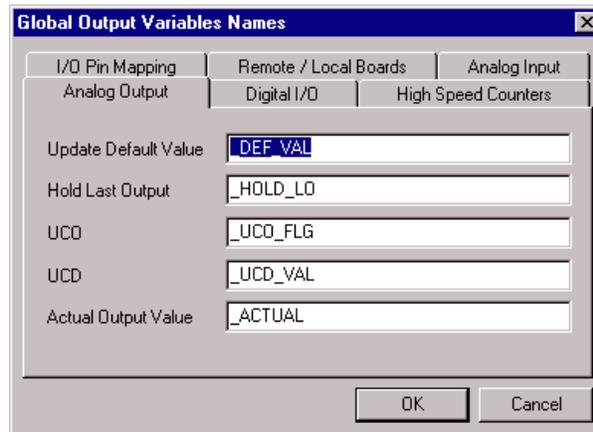
Although the dialog box in the I/O Configurator shows 16 pins, only the first 8 are used; the remaining ones are reserved for future expansion. Pin types are:

Pin Type	Valid for this channel	Notes
Not Configured	Any	The channel is not configured.
Analog Input	Any	Valid only for a single 4-20 mA device on this channel (point-to-point). You must specify a zero and span for the input. Not used with multi-dropped HART devices. If using on Channel 1, set switch SW3 to "IN". If using on Channel 2, set switch SW4 to "IN".
Analog Output	1 or 2	Valid only for a single 4-20 mA HART device on this channel, that is NOT a transmitter. You must specify a zero and span for the output. Not used with multi-dropped HART devices. If using on Channel 1, set switch SW3 to "OUT". If using on Channel 2, set switch SW4 to "OUT".
HART Multi-drop	Any	Up to five multi-dropped HART devices allowed per channel (total of 40 allowed for the entire board – main and daughter). If used on channels 1 or 2, switches SW3 and/or SW4 must be set to "IN".
BTI	Any	Valid for either a Bristol 3508 or Bristol 3808 transmitter. If used on channels 1 or 2, switches SW3 and/or SW4 must be set to "IN".

Changing Default Variable Names (All board types)

As you proceed to define your I/O, the I/O Configuration Wizard will automatically create variable names associated with the I/O board to store status information, zeros and spans, etc.

These variable names are based on the pin name you define with an appropriate suffix added. To see the default suffix, click on the **[Variable Names]** button (previously called **[Settings]**) visible on certain pages of the I/O Configuration Wizard. While NOT recommended, the variable suffixes can be altered by the user, if desired. The different pages of the Global Output Variables Names dialog box are accessible by clicking on the tabs. Make changes on the various pages, then click **[OK]** to save all the changes.



I/O Mapping

Important

Most users **do not** need to be concerned with the details described in this section. The typical user configures I/O using the I/O Configuration Wizard. Only certain users with special I/O requirements or customized software need to be familiar with this information.

In IEC 61131, I/O is addressed by mapping the physical I/O into one of two I/O memory regions. Addresses for these regions are either %Qtxx (Output) or %Itxx (Input), where t is the data type and xx is the offset (for example: %QD100, %QW50, or %QX0.1).

Typically, the inputs are scanned at the start of a task's cycle, and outputs are written at the end. (Note: the user has the choice about which task will process each I/O definition). The user can also force I/O processing to occur by using a standard function block

Common Device Map

The status area is defined for all I/O boards at offset 0 of the input map. This region is 4 bytes long; the first byte is reserved for board errors (see bit definitions in each I/O board section) and the other three are divided as needed by the individual board drivers.

DI

The status of DI points is always mapped from bytes 4-19 of the input map. Any other features (such as counters) will be mapped starting at address offset 20.

DO

The values to be output to a DO are always mapped at bytes 0-15 of the output map. Any other features will be mapped starting at address offset 16.

AI

The AI map is organized into two sections: The input values are mapped as REAL values, starting at offset 8 (there is an exception here when a combo board is used). The output map consists of pairs of REAL values (ZERO, SPAN) by which each input is scaled. The outputs are typically mapped starting at offset 0. If a zero / span pair is not initialized, the scale defaults to ZERO = 0.0 and SPAN = 100.0.

Note:

The Input or Output map may be shortened to reduce the number of points processed.

AO

To output an AO, a set of three REAL values is used: ZERO, SPAN, VALUE. These are mapped starting at offset 0. If a zero / span pair is not initialized, the scale defaults to ZERO = 0.0 and SPAN = 100.0.

Note:

The Input or Output map may be shortened to reduce the number of points processed.

Local I/O - ControlWave

The following sections describe the local I/O boards supported, and their memory maps.

CW_DO32 ControlWave 32 Output Pin Digital Board

DRIVER_NAME: 'CW_DO32'
DATA_TYPE: BYTE
DRIVER_PAR1: Slot number.
DRIVER_PAR2: Bit mask of outputs to be processed by 61131 program. If specified as 0, all points will be allowed.
DRIVER_PAR3: Bit mask for outputs to be processed by 61131 (bits 16-31). If specified as 0, all points will be allowed.
Input Map: Max Size: 8 bytes (6 bytes for 16 point)

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	DO32_x_BOARDSTATUS	Board status. Only bit 0 is currently defined. If set, board is not present.
4-7	DO32_x_y_l	DO status as seen by card. 1 bit per value.

Output Map: Size: 4 bytes (2 bytes on 16 pt)

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	DO32_x_y	Outputs. 1 bit per value. DO1 is LSB; DO8 is MSB. Typically specified as %QXy.z, where y is I/O space offset, and z is bit number from 0 to 7.
1-3	DO32_x_y	Repeat for DO9-32. Offset 1 is DO9-16, 2 is DO17-24, etc.

CW_DI32 ControlWave 32 Input Pin Digital Board

DRIVER_NAME: 'CW_DI32'
 DATA_TYPE: DWORD (32 bits)
 DRIVER_PAR1: Slot number.
 DRIVER_PAR2: Unused
 Input Map: Max Size: 8 bytes (6 bytes for 16 pt)

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	DI32_x_BOARDSTATUS	Board status. Only bit 0 is currently defined. If set, board is not present.
4	DI32_x_y	Current status of DI1 (in bit 0) to DI8 (in bit 7).
5	DI32_x_y	Current status of DI9 (in bit 0) to DI16 (in bit 7).
6	DI32_x_y	Current status of DI17 (in bit 0) to DI24 (in bit 7).
7	DI32_x_y	Current status of DI25 (in bit 0) to DI32 (in bit 7).

CW_AI16 ControlWave 16 Input Pin Analog Board

DRIVER_NAME: 'CW_AI16'
 DATA_TYPE: DWORD (32 bits)
 DRIVER_PAR1: Slot number.
 Input Map: Max Size: 72 bytes (40 for 8 point)

Due to the amount of time required to process the AI points, it is highly recommended that the input region for this board be sized only as large as needed for the points used by the application.

Also, the I/O fetches should be programmed to only occur as fast as needed (via task association).

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0 (Bit 0)	AI16_x_BOARDSTATUS	Board status. Bit 0 is set if board is not present;
0 (Bit 1)	AI16_x_LASTOPERATION	Bit 1 is set if the last conversion operation failed.
4-5	AI16_x_y_OUTRANGE	1 bit per AI, AI1 is bit 0, offset 1, AI8 is bit 7, offset 1, AI9 is bit 0, offset 2. If set, input is Out-of-range.
8	AI16_x_y	Value for AI1 in engineering units (4-byte float - REAL). To access the value, a variable needs to be defined '%IDxx'. Direct access to %IDxx is not possible.
12, 16, ...	AI16_x_y	Value for AI2, AI3, ...

Output Map: Max Size: 128 bytes (64 bytes for 8 point)

To provide consistent scaling values across Application Warm Starts, this I/O region should be marked as 'RETAIN'.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	AI16_x_y_ZERO	Zero for AI1 (4-byte float – REAL). To access the value, define the variable %QDxx. This variable can be initialized at declaration.
4	AI16_x_y_SPAN	Span for AI1 (4-byte float - REAL). If zero, the AI will be scaled from 0 to 100.0.
8, 16, 24, ...	AI16_x_y_ZERO	Zeros for AI2, AI3, ...
12, 20, 28, ...	AI16_x_y_SPAN	Spans for AI2, AI3, ...

CW_AO8 ControlWave 8 Output Pin Analog Board

DRIVER_NAME: 'CW_AO8'

DATA_TYPE: DWORD (32 bits)

DRIVER_PAR1: Slot number.

Input Map: Max Size: 72 bytes (Extra space reserved for future expansion up to AO16)

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0 (Bit 0)	AO8_x_BOARDSTATUS	Board status. Bit 0 is set if board is not present;
0 (Bit 1)	AO8_x_LASTOPERATION	Bit 1 is set if the last conversion operation failed.
4	AO8_x_y_OUTRANGE	1 bit per AO, AO0 is bit 0, AO8 is bit 7. If set, output is Out-of-range.
8, 12, ..., 36	AO8_x_y_ACTUAL	Actual output value for points 1 to 8. If output is Out-of-range low the output will be constrained to the points Zero value. If output is O-o-r high it will be set to the points Zero+Span value.
40, 44, ..., 68	AO8_x_y_ACTUAL	Output values for points 9 to 16 (To allow for future expansion)

Output Map: Max Size: 268 bytes (Extra space reserved for future expansion up to AO16)

Due to the amount of time required to process the AO points, it is highly recommended that the output region for this board be sized only as large as needed for the points used by the application.

Also, the I/O sets should be programmed to only occur as fast as needed (via task association).

To provide consistent scaling values across Application Warm Starts, this I/O region should be marked as 'RETAIN'.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	AO8_x_y_ZERO	Zero for AO1 (4-byte float - REAL). To access the value, define the variable %QDxx. This variable can be initialized at declaration.
4	AO8_x_y_SPAN	Span for AO1 (4-byte float). If zero, the AO will be scaled from 0 to 100.0.
8	AO8_x_y	Value for AO1 (4-byte float).
12, 24, 36, ...	AO8_x_y_ZERO	Zeros for AO2, AO3, AO4, etc.
16, 28, 40, ...	AO8_x_y_SPAN	Spans for AO2, AO3, AO4, etc.
20, 32, 44, ...	AO8_x_y	Values for AO2, AO3, AO4, etc.
192	AO8_x_y_DEF_VAL	Update Default Values for AO1 - AO16 (4-byte integer - DWORD). To access this value, define the variable %QX192.0. This variable can be initialized at declaration. Setting this bit to TRUE will write the default values for AO1 - AO16 to the AO hardware. After the default values are written to the board this variable will then be set back to FALSE.
196	AO8_x_y_HOLD_LO	Hold Last Output (HLO) Control for AO1 - AO16 (4-byte integer - DWORD). To access the values, define variables %QXxxx.x. This variable can be initialized at declaration. (192.0 for AO1, 192.1 for AO2, ... 193.7 for AO16)
200	AO8_x_y_UCO_FLG	User Configured Output (UCO) Control for AO1 - AO16 (4-byte integer - DWORD). To access the value, define variables %QXxxx.x. This variable can be initialized at declaration. (196.0 for AO1, 196.1 for AO2, ... 197.7 for AO16)
204, 208, 212, ...	AO8_x_y_UCD_VAL	User Configured Default (UCD) Value for AO1, AO2, AO3, etc. (4-byte float - REAL). To access the value, define the variable %QDxxx. These variables can be initialized at declaration.

If neither Hold Last State or User Configured Output are enabled, for a point, the output will go to -5% if the unit watchdogs.

If Hold Last State and User Configured Output are both enabled at the same time, for the same point, neither will be the winner. If the unit watchdogs the output will fall back to -5%.

CW_HSC12 ControlWave 12 Channel High Speed Counter / Universal Discrete Input Board

DRIVER_NAME: 'CW_HSC12'
 DATA_TYPE: DWORD (32 bits)
 DRIVER_PAR1: Slot number.

DRIVER_PAR2: UNUSED
Input Map: Max Size: 68 bytes (44 bytes for 6 point)

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	HSC12_x_BOARDSTATUS	Board status. Only bit 0 is currently defined. If set, board is not present.
1	HSC12_x_y_STATE	Input channel state. BOOL Offset 1 Bit 0 is for channel 1, Bit 1 is channel 2, etc. Each bit will reflect the state, either on or off, of the signal on its respective channel.
4	HSC12_x_TIMESTAMP	Timestamp of last sample from HSC. This is the number of milliseconds since boot.
8	HSC12_x_y_COUNTER	Number of counts since boot (Channel 1)
12,16,20, ..., 52	HSC12_x_y_COUNTER	Counts for Channel 2, 3, 4, ..., 12

Output Map: Max Size: 12 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	HSC12_x_y_RESET_COUNT	Reset point counts. BOOL Offset 0, Bit 0 is point 1 ... Offset 1, bit 3 is point 12. Setting a bit to TRUE (1) will reset the count for the point selected. The driver will reset the bit after the count has been reset.
2 (Bit 1)	HSC12_x_NOINIT	Bit 1 – If set to TRUE, maintain counts across warm start.
4	HSC12_x_y_FILTER	30ms/1ms filter select. Offset 4, Bit 0 represent Channel 1 to select 30ms (FALSE) and 1ms (TRUE) respectively. Offset 4, Bit 1 is Channel 2, Offset 5, Bit 3 is Channel 12.
8	HSC12_x_y_HSC_SEL	High Speed Counter select. Offset 8, Bit 0 represent Channel 1 to select High Speed Counter (TRUE). Offset 8, Bit 1 is Channel 2, Offset 9, Bit 3 is Channel 12. Setting an HSC select bit to TRUE will override the 30ms/1ms selection for the same channel.

CW_TC12 – ControlWave 12 Point Thermocouple Board

DRIVER_NAME 'CW_TC12'
DATA_TYPE DWORD (32 bits)
DRIVER_PAR1 slot number.
Input Map: Max Size: 56 bytes

Due to the amount of time required to process the thermocouple points, it is highly recommended that this I/O driver be assigned to a task (instead of “No Task”). Care should also be taken in using the I/O board in a task of less than 40ms.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0 (Bit 0)	TC12_x_BOARDSTATUS	If set, the board is not present.
0 (Bit 2)	TC12_x_CALIBRATE	If set indicates invalid calibration data written to the board.
0 (Bit 3)	TC12_x_TIMEOUT	If set, indicates that had an error reading or writing to the board.
4	TC12_x_y_OUTRANGE	1 bit per TC, TC1 is bit 0, TC8 is bit 7. If set, input is Out-of-range.
5	TC12_x_y_OUTRANGE	1 bit per TC, TC9 is bit 0, TC12 is bit 3. If set, input is Out-of-range.
8	TC12_x_y	Value for TC1 in engineering units (4-byte float - REAL). To access the value, define the variable AT %IDxx. Direct access to %IDxx is not possible.
12, 16, 20, 24, ... 52	TC12_x_y	Value for TC2, TC3, TC4, TC5, TC6... TC12

Output Map: Max Size: 112 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	TC12_x_y_ZERO	Zero for TC1 (4-byte float - REAL). To access the value, define the variable AT %QDxx. This variable can be initialized at declaration.
4	TC12_x_y_SPAN	Span for TC1 (4-byte float). If zero, the TC will be scaled as in the chart below. If specified, the new value will be $ORG_VALUE * Span + Zero$.
8, 16, 24, 32, ... 88	TC12_x_y_ZERO	Zeros for TC2, TC3, TC4, to TC12 – Example: for C to F, use 32.0
12, 20, 28, 36, ... 92	TC12_x_y_SPAN	Spans for TC2, TC3, TC4, to TC12 – Example for C to F, use 1.8
100	TC12_x_y_MODE	Point type for TC1; see Thermocouple type codes section for details.
101, 102, 103, 104, ... 111	TC12_x_y_MODE	Point types for TC2, TC3, TC4, to TC12.

Type codes for Thermocouple Points.

Type Code	Code	Range
0	B	Thermocouple: 100C – +1820C
1	E	Thermocouple: -270C – +1000C
2	J	Thermocouple: -210C – +1200C
3	K	Thermocouple: -270C – +1370C
4	R	Thermocouple: -50C – +1720C
5	S	Thermocouple: -50C – +1760C

Type Code	Code	Range
6	T	Thermocouple: -270C – +400C
7	Unused	Unused
8	10MV	Voltage Inputs: -10 mV to +10 mV (Outputs as 0.0 to 1.0)
9	C	Thermocouple: 0C – +2315C
10	N	Thermocouple: -270C – +1300C

CW_RTD8 - ControlWave 8 Point Resistance Temperature Device (RTD) Board

DRIVER_NAME 'CW_RTD8'
 DATA_TYPE DWORD (32 bits)
 DRIVER_PAR1 slot number.
 Input Map: Max Size: 40 bytes

Due to the amount of time required to process the RTD points, it is highly recommended that this I/O driver be assigned to a task (instead of "No Task"). Care should also be taken in using the I/O board in a task of less than 40ms.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0 (Bit 0)	RTD8_x_BOARDSTATUS	If set, the board is not present.
0 (Bit 2)	RTD8_x_CALIBRATE	If set indicates invalid calibration data written to the board.
0 (Bit 3)	RTD8_x_TIMEOUT	If set, indicates that had an error reading or writing to the board.
1 (Bit 0)	RTD8_x_LASTCALBOP	Set if last calibration or reset operation failed.
1 (Bit 7)	RTD8_x_CALBCMD	Calibration Commands Allowed. Until this bit is set, all calibration commands are ignored.
4	RTD8_x_y_READERR	RTD Reading Error. Bit 0 is RTD1, Bit 7 is RTD8
8	RTD8_x_y	RTD1 reading – REAL – In units of Degrees Centigrade (unless scaled by values in the output map).
12, 16, ...36	RTD8_x_y	Readings for RTD2 ... RTD8.

Output Map: Max Size: 280 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RTD8_x_y_ZERO	Zero for RTD 1(4-byte float - REAL). Example: for C to F, use 32.0. Defaults to 0.0
4	RTD8_x_y_SPAN	Span for RTD 1(4-byte float – REAL). If zero, RTD will not be scaled. If specified, the scaled value will be $ORG_VALUE * Span + Zero$. Example for C to F, use 1.8.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
8,16,24,32 40,48,56	RTD8_x_y_ZERO	Zero for RTDs 2-8
12,20,28, 36,44,52, 60	RTD8_x_y_SPAN	Span for RTDs 2-8
120 (Bit 0) *	RTD8_x_y_RESTORE	If set, restore RTD 1 calibration to Factory Defaults. Will be reset when operation completes.
121 *	RTD8_x_y_OPERATION	Calibration Operation for RTD 1 – SINT 5 RTD Zero (100 Ohms) 6 RTD Span (300 Ohms) 7 RTD Coefficients (A, B, R0) 8 RTD Span (not using 300 Ohms)
124 **	RTD8_x_y_COEFF_A	Coefficient A (RTD 1)
128 **	RTD8_x_y_COEFF_B	Coefficient B (RTD 1)
132 **	RTD8_x_y_COEFF_R0	Coefficient R0 (RTD 1)
136 **	RTD8_x_y_APPLIED	The applied temperature when calibration operation 8 was performed. (RTD 1)
140 (Bit 0) *	RTD8_x_y_RESTORE	If set, restore RTD 2 calibration to Factory Defaults. Will be reset when operation completes.
141 *	RTD8_x_y_OPERATION	Calibration Operation for RTD 2 – SINT 5 RTD Zero (100 Ohms) 6 RTD Span (300 Ohms) 7 RTD Coefficients (A, B, R0) 8 RTD Span (not using 300 Ohms)
144 **	RTD8_x_y_COEFF_A	Coefficient A (RTD 2)
148 **	RTD8_x_y_COEFF_B	Coefficient B (RTD 2)
152 **	RTD8_x_y_COEFF_R0	Coefficient R0 (RTD 2)
156 **	RTD8_x_y_APPLIED	The applied temperature when calibration operation 8 was performed. (RTD 2)
....	
260 (Bit 0) *	RTD8_x_y_RESTORE	If set, restore RTD 8 calibration to Factory Defaults. Will be reset when operation completes.
261 *	RTD8_x_y_OPERATION	Calibration Operation for RTD 8 – SINT 5 RTD Zero (100 Ohms) 6 RTD Span (300 Ohms) 7 RTD Coefficients (A, B, R0) 8 RTD Span (not using 300 Ohms)
264*	RTD8_x_y_COEFF_A	Coefficient A (RTD 8)
268**	RTD8_x_y_COEFF_B	Coefficient B (RTD 8)
272**	RTD8_x_y_COEFF_R0	Coefficient R0 (RTD 8)
276**	RTD8_x_y_APPLIED	The applied temperature when calibration

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
		operation 8 was performed. (RTD 8)

- * Value written to perform operation. The value will be reset by driver when the operation completes.
- ** Value is read from the board by the driver. In order to perform calibration operations 7 and 8, the user can overwrite the values; then, issue the calibration command.

Ethernet I/O

Data from Ethernet I/O units will be transferred to the process automation controller using a TCP/IP data link. The data link may be a dedicated Ethernet line or may have other IP traffic. Performance-critical applications should not be run in over a shared link.

A series of special I/O configuration modules, *boards*, will be defined to support Ethernet I/O. These modules are to be included by the application developer and a firmware driver will support each module. The drivers will be linked as a part of the system firmware for the controller. These drivers will provide a *front-end* to a common driver that hands TCP/IP communication, Modbus mapping and timing responsibilities. This driver will be responsible for exchanging data between the I/O configuration memory and the actual Ethernet I/O hardware units.

A special software program for Ethernet I/O configuration, called the **Remote I/O Toolkit**, is included as an installation option on the OpenBSI CD ROM.

A copy of the I/O image memory that supports the Ethernet I/O data will be held in a buffer controlled by the common driver. This driver will copy the data into the I/O image memory whenever a call is made to the driver's **read** member function. A read call will also initiate a communication request to refresh the image data from the remote hardware. Similarly, the I/O image memory will be copied into the driver's image buffer when the **write** member function of the driver is called.

Sending the data request message will be delayed, such that the response data will arrive just before the next scheduled driver read call. The driver will collect statistics, possibly a rolling average, in order to calculate the optimal amount of time to delay. Problems could occur in this optimal time calculation if either the request for data is not regular or the data link is not very consistent in response time.

To provide some indication of the freshness of the Ethernet I/O data, a *read request counter* is included in each input I/O configuration block. This counter will be incremented during the Read input portion of application task's execution. The increment will occur just after the image data is copied. The driver will clear this count while updating its internal image from a data response message. This way if a fresh block of data is received the counter will be zero, but if the data is the same as the last read request, the counter will be greater than zero.

The Open Modbus standard is supported using TCP/IP. The units contain an ID string, which will be used to verify that the proper unit type is mapped to an IP address. No other verification of the Ethernet I/O configuration will be performed. It is assumed that this function will be done using the Remote I/O Toolkit.

Also implemented are two general-purpose Open Modbus I/O boards to allow communication with other compliant hardware without requiring new firmware. One board will support input and the other output.

Driver status for all board types is defined as follows:

Bit	Bit Value	Meaning
0	1	Cannot communicate properly with Ethernet I/O module.
1	2	Communications can be established with the Ethernet I/O device, but, the data response cannot be properly parsed. This is most likely due to an invalid device type being configured.
3	8	Configuration Error. For units which support a device ID request, if this is set, the unit responded with an unexpected unit identifier.

The memory map of each module is described below.

BB_8DI8DO (8 Remote Digital Input and 8 Remote Digital Output Pin Ethernet I/O Board)

DRIVER_NAME: 'BB_8DI8DO'

DATA_TYPE: BYTE

DRIVER_PAR1: always 0

DRIVER_PAR2 : IP address - first and second *quads*.

DRIVER_PAR3: IP address - third and fourth *quads*

DRIVER_PAR4: Open Modbus Unit number - used for gateway chaining.

Input Map: Max Size: 52 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RDIDO_x_DRIVERSTATUS	Driver status - see table above.
1	RDIDO_x_FRESHNESSCOUNT	Freshness counter.
4	RDIDO_x_y	Current status of DI1 (in bit 0) to DI8 (in bit 7).
5-19		spare
20/23	RDIDO_x_y_COUNTER	If counter processing is enabled, the number of counts for DI1. A count is registered when the DI transitions from low to high. Note: Only a 16 bit counter is held, but a 32 bit value will be reported here. Roll over in the 16 bit counter will be carried into the 32-bit counts.
24/27, ..., 48/51	RDIDO_x_y_COUNTER	Counts for DI2, ..., DI8.

Output Map: Size: 48 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RDIDO_x_y_O	Outputs. 1 bit per value. DO1 is LSB; DO8 is MSB. Typically specified as %QXy.z, where y is I/O space offset, and z is bit number from 0 to 7.
1-15		Spare
16	RDIDO_x_ACTIVATECOUNTER	Activate counters. 1 bit per point.
17	RDIDO_x_CLEARCOUNTER	Clear counters. Sets values back to 0. 1 bit per point.
18/19	RDIDO_x_y_O_TPOVALUE	TPO value for DO point 1. 1 word / DO point for the Pulse count in the range 0 to 32767. See 'Tpo' and the 'Time Proportioned Output' topic in the Remote I/O Tool Kit on-line help for more information.
20/21, ..., 46/47	RDIDO_x_y_O_TPOVALUE	TPO value for DO points 2 ... 8.

Modbus Function Codes: 02 to read DI values @ 1 ... 8
 04 to read Counter values @ 1 ... 8
 0F_n to set DO values @ 1 ... 8
 10_n to set TPO values @ 1 ... 8

BB_16DI-(16 Remote Digital Input Pin Ethernet I/O Board)

DRIVER_NAME: 'BB_16DI'
DRIVER_PAR1: always 0
DRIVER_PAR2: IP address - first and second *quads*.
DRIVER_PAR3: IP address - third and fourth *quads*
DRIVER_PAR4: Open Modbus Unit number - used for gateway chaining.

Input Map: Max Size: 84 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RDI_x_DRIVERSTATUS	Driver status - see table above.
1	RDI_x_FRESHNESSCOUNT	Freshness counter.
4	RDI_x_y	Current status of DI1 (in bit 0) to DI8 (in bit 7).
5	RDI_x_y	Current status of DI9 (in bit 0) to DI16 (in bit 7).
6-19		Spare
20/23	RDI_x_y_COUNTER	If counter processing is enabled, the number of counts for DI1. A count is registered when the DI transitions from low to high. Note: Only a 16-bit counter is held, but a 32-bit value will be reported here. Roll over in the 16-bit counter will be carried

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
		into the 32-bit counts.
24/27, ..., 80/83	RDI_x_y_COUNTER	Counts for DI2, ..., DI16.

Output Map:

Size: 4 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0/1	RDI_x_ACTIVATECOUNTER	Activate counters. 1 bit per point. Counters 1-8 in first byte, 9-16 in second.
2/3	RDI_x_CLEARCOUNTER	Clear counters. Sets values back to 0. 1 bit per point.

Modbus Function Codes: 02 to read DI values @ 1 ... 16

04 to read Counter values @ 1 ... 16

BB_16DO (16 Remote Digital Output Pin Ethernet I/O Board)

DRIVER_NAME: 'BB_16DO'

DATA_TYPE: BYTE

DRIVER_PAR1: always 0

DRIVER_PAR2: IP address - first and second *quads*.DRIVER_PAR3: IP address - third and fourth *quads*

DRIVER_PAR4: Open Modbus Unit number - used for gateway chaining.

Input Map: Max Size: 2 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RDO_x_DRIVERSTATUS	Driver status - see table above.
1	RDO_x_FRESHNESSCOUNT	Freshness counter.

Output Map:

Size: 48 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RDO_x_y	Outputs. 1 bit per value. DO1 is LSB; DO8 is MSB. Typically specified as %QXy.z, where y is I/O space offset, and z is bit number from 0 to 7.
1	RDO_x_y	Outputs. 1 bit per value. DO9 is LSB; DO16 is MSB.
2-15		spare
16/17	RDO_x_y_TPOVALUE	TPO value for DO point 1. 1 word / DO point for the Pulse count in the range 0 to 32767. See 'Tpo' and the 'Time Proportioned Output' topic in the Remote I/O

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
		Tool Kit on-line help for more information.
18/19, ..., 46/47	RDO_x_y_TPOVALUE	TPO for DO points 2 to 16.

Modbus Function Codes: 0F_n to set DO values @ 1 ... 16

 10_n to set TPO values @ 1 ... 16

BB_8DI8AI - (8 Remote Digital Input and 8 Remote Analog Input Pin Ethernet I/O Board)

Note:

For Proper functioning of the AIs, the Remote I/O Toolkit must be used to set the "Features" for each channel to be "Positive Only" or "- Below 4mA".

DRIVER_NAME: 'BB_8DI8AI'

DATA_TYPE: BYTE

DRIVER_PAR1: always 0

DRIVER_PAR2: IP address - first and second *quads*.

DRIVER_PAR3: IP address - third and fourth *quads*

DRIVER_PAR4: Open Modbus Unit number - used for gateway chaining.

Input Map: Max Size: 84 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RDIAI_x_DRIVERSTATUS	Driver status - see table above.
1	RDIAI_x_FRESHNESSCOUNT	Freshness counter.
4	RDIAI_x_y_DI	Current status of DI1 (in bit 0) to DI8 (in bit 7).
5-19		Spare
20/23	RDIAI_x_y_DI_COUNTER	If counter processing is enabled, the number of counts for DI1. A count is registered when the DI transitions from low to high.
24/27, ..., 48/51	RDIAI_x_y_DI_COUNTER	Counts for DI2, ..., DI8.
52/55	RDIAI_x_y_AI	Value for AI1 in engineering units (4-byte float - REAL). To access the value, define the variable %IDxx. Direct access to %IDxx is not possible.
56/59, ..., 80/83	RDIAI_x_y_AI	Values for AI2 to AI8.

Output Map: Size: 68 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RDIAI_x_ACTIVATECOUNTER	Activate counters. 1 bit per point. Typically specified as %QXy.z, where y is I/O space offset, and z is bit number from 0 to 7.
1	RDIAI_x_CLEARCOUNTER	Clear counters. Sets values back to 0. 1 bit per point.
4/7	RDIAI_x_y_AI_ZERO	Zero for AI1 (4-byte float - REAL). To access the value, define the variable %QDxx. This variable can be initialized at declaration.
8/11	RDIAI_x_y_AI_SPAN	Span for AI1 (4-byte float). If zero, the AI will be scaled from 0 to 100.0.
12/15	RDIAI_x_y_AI_ZERO	Zero for AI2.
16/19	RDIAI_x_y_AI_SPAN	Span for AI2.
20/23, ..., 64/67		Zeros and Spans for AI3 ... AI8.

Modbus Function Codes: 02 to read DI values @ 1 ... 8
 04 to read Counter and AI values @ 1 ... 8
 0F_h to set /clear counters @ 1 ... 8

BB_16AI (16 Remote Analog Input Pin Ethernet I/O Board)

Note:

For Proper functioning of the AIs, the Remote I/O Toolkit must be used to set the "Features" for each channel to be "Positive Only" or "- Below 4mA".

DRIVER_NAME: 'BB_16AI'
 DATA_TYPE: DWORD (32 bits)
 DRIVER_PAR1: always 0
 DRIVER_PAR2: IP address - first and second *quads*.
 DRIVER_PAR3: IP address - third and fourth *quads*
 DRIVER_PAR4: Open Modbus Unit number - used for gateway chaining.
 Input Map: Max Size: 68 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RAI_x_DRIVERSTATUS	Driver status - see table above.
1	RAI_x_FRESHNESSCOUNT	Freshness counter.
2/3		Spare

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
4/7	RAI_x_y	Value for AI1 in engineering units (4-byte float - REAL). To access the value, define the variable %Dxx. Direct access to %IDxx is not possible.
8/11	RAI_x_y	Value for AI2
12/15, ..., 64/67	RAI_x_y	Values for AI3 to AI16.

Output Map: Max Size: 128 bytes per slot

To provide consistent scaling values across Application Warm Starts, this I/O region should be marked as **RETAIN**.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0/3	RAI_x_y_ZERO	Zero for AI1 (4-byte float - REAL). To access the value, define the variable %QDxx. This variable can be initialized at declaration.
4/7	RAI_x_y_SPAN	Span for AI1 (4-byte float). If zero, the AI will be scaled from 0 to 100.0.
8/11	RAI_x_y_ZERO	Zero for AI2.
12/15	RAI_x_y_SPAN	Span for AI2.
16/19 ... 124/127		Zeros and Spans for AI3 ... AI16.

Modbus Function Codes: 04 to read AI values @ 1 ... 16

BB_8AI4AO - (8 Remote Analog Input and 4 Remote Analog Output Pin Ethernet I/O Board)

Note:

Note: For Proper functioning of the AIs, the Remote I/O Toolkit must be used to set the "Features" for each channel to be "Positive Only" or "- Below 4mA".

```

DRIVER_NAME:      'BB_8AI4AO'
DATA_TYPE:        DWORD                (32 bits)
DRIVER_PAR1:      always 0
DRIVER_PAR2:      IP address - first and second quads.
DRIVER_PAR3:      IP address - third and fourth quads
DRIVER_PAR4:      Open Modbus Unit number - used for gateway chaining.
Input Map:        Max Size: 36 bytes

```

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RAIAO_x_DRIVERSTATUS	Driver status - see table above.
1	RAIAO_x_FRESHNESSCOUNT	Freshness counter.
2/3		spare
4/7	RAIAO_x_y	Value for AI1 in engineering units (4-byte float - REAL). To access the value, define the variable %IDxx. Direct access to %IDxx is not possible.
8/11, ..., 32/35	RAIAO_x_y	Values for AI2 to AI8.

Output Map: Max Size: 112 bytes

To provide consistent scaling values across Application Warm Starts, this I/O region should be marked as **RETAIN**.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0/3	RAIAO_x_y_ZERO	Zero for AI1 (4-byte float - REAL). To access the value, define the variable %QDxx. This variable can be initialized at declaration.
4/7	RAIAO_x_y_SPAN	Span for AI1 (4-byte float). If zero, the AI will be scaled from 0 to 100.0.
8/11, ..., 28/31		Zeros and Spans for AI2 ... AI8.
64/67	RAIAO_x_y_O_ZERO	Zero for AO1 (4-byte float - REAL). To access the value, define the variable %QDxx. This variable can be initialized at declaration.
68/71	RAIAO_x_y_O_SPAN	Span for AO1 (4-byte float). If zero, the AO will be scaled from 0 to 100.0.
72/75	RAIAO_x_y_O	Value for AO1 (4-byte float).
76/79, , 108/111		Zeros, Spans and Values for AO2 to AO4.

Modbus Function Codes: 04 to read AI values @ 1 ... 8

10_h to set AO value @ 1 ... 4

BB_8INS (Instrumentation Ethernet I/O Board)

DRIVER_NAME: 'BB_8INS'
 DATA_TYPE: DWORD (32 bits)
 DRIVER_PAR1: always 0
 DRIVER_PAR2: IP address - first and second *quads*.
 DRIVER_PAR3: IP address - third and fourth *quads*

DRIVER_PAR4: Open Modbus Unit number - used for gateway chaining.

Input Map: Max Size: 36 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RIN_x_DRIVERSTATUS	Driver status - see table above.
1	RIN_x_FRESHNESSCOUNT	Freshness counter.
2/3		Spare
4/7	RIN_x_y	Value for AI1 in engineering units (4-byte float - REAL). To access the value, define the variable %IDxx. Direct access to %IDxx is not possible.
8/11, ..., 32/35	RIN_x_y	Values for AI2 to AI8.

Output Map: Max Size: 32 bytes

To provide consistent scaling values across Application Warm Starts, this I/O region should be marked as **RETAIN**.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0/3	RIN_x_y_ZERO	Zero for AI1 (4-byte float - REAL). To access the value, define the variable %QDxx. This variable can be initialized at declaration.
4/7	RIN_x_y_SPAN	Span for AI1 (4-byte float). If zero, the AI will be scaled from 0 to 100.0.
8/11, ..., 28/31		Zeros and Spans for AI2 ... AI8.

Modbus Function Codes: 04 to read AI values @ 1 ... 8

BB_8HSC-(8 Channel High Speed Counter Channel Ethernet I/O Board)

Note:

For proper functioning of the counters, the Remote I/O Toolkit must be used to set up for 32-bit counters. Also, the counts reported are the RAW counts from the module. No adjustment is performed to convert to counts since boot.

DRIVER_NAME: 'BB_8HSC'

DATA_TYPE: DWORD (32 bits)

DRIVER_PAR1: always 0

DRIVER_PAR2: IP address - first and second *quads*.

DRIVER_PAR3: IP address - third and fourth *quads*
 DRIVER_PAR4: Open Modbus Unit number - used for gateway chaining.
 Input Map: Max Size: 40 bytes

Offset	Default Variable Name where <i>x</i> is the board slot and <i>y</i> is the pin number.	Description
0	RHSC_x_DRIVERSTATUS	Driver status - see table above.
1	RHSC_x_FRESHNESSCOUNT	Freshness counter.
2/3		Spare
4	RHSC_x_TIMESTAMP	Timestamp of last sample from HSC. This is the number of milliseconds since boot.
8	RHSC_x_y_COUNTER	Number of counts since boot (Channel 1)
12, 16, ..., 36	RHSC_x_y_COUNTER	Counts for Channel 2, 3, ..., 8

Modbus Function Codes: 04 to read HSC value @ 1 ... 8 (must read 2 16-bit register pairs per counter)

IPMB_INP (Open Modbus – Input Ethernet I/O Board)

DRIVER_NAME: 'IPMB_INP'
 DATA_TYPE: BYTE
 DRIVER_PAR1: Low Byte – Function Code (see below) - High byte undefined
 DRIVER_PAR2: IP address - first and second *quads*.
 DRIVER_PAR3: IP address - third and fourth *quads*
 DRIVER_PAR4: Modbus unit number.

Function Code:	Function Description:	Number of Bytes in Response: (N=requested elements)
01	Read Coil Status	N / 8
02	Read Input Status	N / 8
03	Read Holding Registers	2 * N
04	Read Input Registers	2 * N

Input Map: Max Size: (2 + requested size) bytes

Offset	Default Variable Name where <i>x</i> is the board slot and <i>y</i> is the pin number.	Description
0	RIN_x_DRIVERSTATUS	Driver status - see table above.
1	RIN_x_FRESHNESSCOUNT	Freshness counter.
4, ...		See function codes for size of responses in bytes - above.

Output Map: Max Size: 6

Offset	Default Variable Name where <i>x</i> is the board slot and <i>y</i> is the pin number.	Description
0,1	RIN_x_STARTREGISTER	Modbus Register/Coil start address
2,3	RIN_x_NUMREGISTERS	Number of Registers or Coils.

Note:

The fields in the output map are only settable one time. Once a non-zero number is written to Offset 2,3, the request is defined and will start collecting.

IPMB_OUT (Open Modbus – Output - Ethernet I/O Board)

DRIVER_NAME: 'IPMB_OUT'
 DATA_TYPE: BYTE
 DRIVER_PAR1: Low Byte – Function Code (see below) - High byte undefined
 DRIVER_PAR2: IP address - first and second *quads*.
 DRIVER_PAR3 : IP address - third and fourth *quads*
 DRIVER_PAR4: Modbus unit number.

Function Code:	Function Description:	Number of Bytes in Response: (N=requested elements)
0F	Force Multiple Coils	N / 8
10	Set Multiple Registers	2 * N

Input Map: Max Size: 2 bytes

Offset	Default Variable Name where <i>x</i> is the board slot and <i>y</i> is the pin number.	Description
0	ROUT_x_DRIVERSTATUS	Driver status - see table above.
1	ROUT_x_FRESHNESSCOUNT	Freshness counter.

Output Map: Max Size: 6 + data size - see above.

Offset	Default Variable Name where <i>x</i> is the board slot and <i>y</i> is the pin number.	Description
0,1	ROUT_x_STARTREGISTER	Modbus Register/Coil start address
2,3	ROUT_x_NUMREGISTERS	Number of Registers or Coils.
4, ...		Data bytes to be output.

BB_4RTDI (RTD - Resistance Temperature Device Ethernet I/O Board)**Note:**

For proper functioning of the RTDs, the Remote I/O Toolkit must be used to set the proper scaling range.

DRIVER_NAME: 'BB_4RTDI'
 DATA_TYPE: BYTE
 DRIVER_PAR1: always 0
 DRIVER_PAR2: IP address - first and second *quads*.
 DRIVER_PAR3: IP address - third and fourth *quads*
 DRIVER_PAR4: Open Modbus Unit number - used for gateway chaining.
 Input Map: Max Size: 52 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RTDDI_x_DRIVERSTATUS	Driver status - see table above.
1	RTDDI_x_FRESHNESSCOUNT	Freshness counter.
2-3		Spare
4	RTDDI_x_y_DI	Current status of DI1 (in bit 0) to DI4 (in bit 4).
5-19		Spare (for driver consistency)
20-23	RTDDI_x_y_DI_COUNTER	If counter processing is enabled, the number of counts for DI1. A count is registered when the DI transitions from low to high.
24-27	RTDDI_x_y_DI_COUNTER	Counts for DI2
28-31	RTDDI_x_y_DI_COUNTER	Counts for DI3
32-35	RTDDI_x_y_DI_COUNTER	Counts for DI4
36-39	RTDDI_x_y_AI	Value for RTD1 in engineering units (4-byte float - REAL). To access the value, define the variable %ldxx. Direct access to %ldxx is not possible.
40-43	RTDDI_x_y_AI	Value for RTD2
44-47	RTDDI_x_y_AI	Value for RTD3
48-51	RTDDI_x_y_AI	Value for RTD4

Output Map: Size: 3 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RTDDI_x_ACTIVATECOUNTER	Activate counters. 1 bit per point. Typically specified as %Qby.z, where y is I/O space offset, and z is bit number from 0 to 3.
1	RTDDI_x_CLEARCOUNTER	Clear counters. Set values back to 0. 1 bit per point.
2.0	RTDDI_x_DISPLAYTENTHS	Convert RTD values to tenths precision. Unit level control. True = tenths (BOOL).

Modbus Function Codes: 02 to read DI values @ 1 ... 4
 04 to read Counter and AI values @ 1 ... 4
 0F_n to set / clear counters @ 1 ... 4

ControlWave I/O Expansion Rack Boards

Common Status Information

The first two bytes of the input map contain status information, which is common to all Expansion Rack boards.

Byte	Bit	Description
0	0 (0x1, 1)	No board is present in the destination rack.
0	3 (0x8, 8)	Board type does not match the board installed in the rack.
0	4 (0x10, 16)	Communications lost with the expansion rack.
0	7 (0x80, 128)	Initial opening of channel to the expansion rack has not been completed.

Driver status for each I/O Expansion Rack board type is stored as an USINT.

ER_DO32 32 Digital Output Pin ControlWave I/O Expansion Rack Board

DRIVER_NAME: 'ER_DO32'
DATA_TYPE: BYTE
DRIVER_PAR1: Slot number
DRIVER_PAR2: IP address - first half
DRIVER_PAR3: IP address - second half
DRIVER_PAR4: Specifies redundant board (0 = False, 1 = True)
Input Map: Max Size: 8 bytes (6 bytes for 16 point)

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERDO_x_DRIVERSTATUS	Board Status (see Common Status Information section)
4-7	ERDO_x_y_I	DO status as seen by card. 1 bit per value.

Output Map: Size: 4 bytes (2 bytes on 16 pt)

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERDO_x_y	Outputs. 1 bit per value. DO1 is LSB; DO8 is MSB. Typically specified as %QXy.z, where y is I/O space offset, and z is bit number from 0 to 7.
1-3	ERDO_x_y	Repeat for DO9-32. Offset 1 is DO9-16, 2 is DO17-24, etc.

ER_DI32 32 Digital Input Pin ControlWave I/O Expansion Rack Board

DRIVER_NAME: 'ER_DI32'
 DATA_TYPE: DWORD (32 bits)
 DRIVER_PAR1: Slot number
 DRIVER_PAR2: IP address - first half
 DRIVER_PAR3: IP address - second half
 DRIVER_PAR4: Specifies redundant board (0 = False, 1 = True)
 Input Map: Max Size: 8 bytes (6 bytes for 16 pt)

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERDI_x_DRIVERSTATUS	Board status. (See Common Status Information section)
4	ERDI_x_y	Current status of DI1 (in bit 0) to DI8 (in bit 7).
5	ERDI_x_y	Current status of DI9 (in bit 0) to DI16 (in bit 7).
6	ERDI_x_y	Current status of DI17 (in bit 0) to DI24 (in bit 7).
7	ERDI_x_y	Current status of DI25 (in bit 0) to DI32 (in bit 7).

ER_AI16 16 Analog Input Pin ControlWave I/O Expansion Rack Board

DRIVER_NAME: 'ER_AI16'
 DATA_TYPE: DWORD (32 bits)
 DRIVER_PAR1: Slot number
 DRIVER_PAR2: IP address - first half
 DRIVER_PAR3: IP address - second half
 DRIVER_PAR4: Specifies redundant board (0 = False, 1 = True)
 Input Map: Max Size: 72 bytes (40 for 8 point)

Due to the amount of time required to process the AI points, it is highly recommended that the input region for this board be sized only as large as needed for the points used by the application.

Also, the I/O fetches should be programmed to only occur as fast as needed (via task association).

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERAI_x_DRIVERSTATUS	Board status. (see Common Status Information section)
2 (bit 1)	ERAI_x_LASTOPERATION	Bit 1 is set if the last conversion operation failed.
4-5	ERAI_x_y_OUTRANGE	1 bit per AI, AI1 is bit 0, offset 1, AI8 is bit 7, offset 1, AI9 is bit 0, offset 2. If set, input is Out-of-range.
8	ERAI_x_y	Value for AI1 in engineering units (4-byte float - REAL). To access the value, define the variable %IDxx. Direct access to %IDxx is not possible.
12, 16, ...	ERAI_x_y	Value for AI2, AI3, ...

Output Map: Max Size: 128 bytes (64 bytes for 8 point)

To provide consistent scaling values across Application Warm Starts, this I/O region should be marked as 'RETAIN'.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERAI_x_y_ZERO	Zero for AI1 (4-byte float - REAL). To access the value, define the variable %QDxx. This variable can be initialized at declaration.
4	ERAI_x_y_SPAN	Span for AI1 (4-byte float - REAL). If zero, the AI will be scaled from 0 to 100.0.
8, 16, 24, ...	ERAI_x_y_ZERO	Zeros for AI2, AI3, ...
12, 20, 28, ...	ERAI_x_y_SPAN	Spans for AI2, AI3, ...

ER_AO8 8 Analog Output Pin ControlWave I/O Expansion Rack Board

DRIVER_NAME:	'ER_AO8'
DATA_TYPE:	DWORD (32 bits)
DRIVER_PAR1:	Slot number
DRIVER_PAR2:	IP address - first half
DRIVER_PAR3:	IP address - second half
DRIVER_PAR4:	Specifies redundant board (0 = False, 1 = True)

Input Map: Max Size: 72 bytes (Extra space reserved for future expansion up to AO16)

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERAO_x_DRIVERSTATUS	Board status. (See Common Status Information section)
2 (bit 1)	ERAO_x_LASTOPERATION	Bit 1 is set if the last conversion operation failed.
4	ERAO_x_y_OUTRANGE	1 bit per AO, AO0 is bit 0, AO8 is bit 7. If set, output is Out-of-range.
8, 12, ..., 36	ERAO_x_y_ACTUAL	Actual output value for points 1 to 8. If output is out-of-range low the output will be constrained to the points Zero value. If output is 0-or high it will be set to the points Zero+Span value.
40, 44, ..., 68	ERAO_x_y_ACTUAL	Output values for points 9 to 16 (To allow for future expansion)

Output Map: Max Size: 268 bytes (Extra space reserved for future expansion up to AO16)

Due to the amount of time required to process the AO points, it is highly recommended that the output region for this board be sized only as large as needed for the points used by the application.

Also, the I/O sets should be programmed to only occur as fast as needed (via task association).

To provide consistent scaling values across Application Warm Starts, this I/O region should be marked as 'RETAIN'.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERAO_x_y_ZERO	Zero for AO1 (4-byte float - REAL). To access the value, define the variable %QDxx. This variable can be initialized at declaration.
4	ERAO_x_y_SPAN	Span for AO1 (4-byte float). If zero, the AO will be scaled from 0 to 100.0.
8	ERAO_x_y	Value for AO1 (4-byte float).
12, 24, 36, ...	ERAO_x_y_ZERO	Zeros for AO2, AO3, AO4, etc.
16, 28, 40, ...	ERAO_x_y_SPAN	Spans for AO2, AO3, AO4, etc.
20, 32, 44, ...	ERAO_x_y	Values for AO2, AO3, AO4, etc.
192	ERAO_x_y_DEF_VAL	Update Default Values for AO1 - AO16 (4-byte integer - DWORD). To access this value, define the variable %QX192.0. This variable can be initialized at declaration. Setting this bit to TRUE will write the default values for AO1 - AO16 to the AO hardware. After the default values are written to the board this variable will then be set back to FALSE.
196	ERAO_x_y_HOLD_LO	Hold Last Output (HLO) Control for AO1 - AO16

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
		(4-byte integer - DWORD). To access the values, define variables %QXxxx.x. This variable can be initialized at declaration. (192.0 for AO1, 192.1 for AO2, ... 193.7 for AO16)
200	ERAO_x_y_UCO_FLG	User Configured Output (UCO) Control for AO1 - AO16 (4-byte integer - DWORD). To access the value, define variables %QXxxx.x. This variable can be initialized at declaration. (196.0 for AO1, 196.1 for AO2, ... 197.7 for AO16)
204, 208, 212, ...	ERAO_x_y_UCD_VAL	User Configured Default (UCD) Value for AO1, AO2, AO3, etc. (4-byte float - REAL). To access the value, define the variable %QDxxx. These variables can be initialized at declaration.

If neither Hold Last State or User Configured Output are enabled, for a point, the output will go to -5% if the unit watchdogs.

If Hold Last State and User Configured Output are both enabled at the same time, for the same point, neither will be the winner. If the unit watchdogs the output will fall back to -5%.

ER_HSC12 12 Channel High Speed Counter ControlWave I/O Expansion Rack Board

DRIVER_NAME: 'ER_HSC12'
 DATA_TYPE: DWORD (32 bits)
 DRIVER_PAR1: Slot number
 DRIVER_PAR2: IP address - first half
 DRIVER_PAR3: IP address - second half
 DRIVER_PAR4: Specifies redundant board (0 = False, 1 = True)
 Input Map: Max Size: 68 bytes (44 bytes for 6 point)

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERHSC_x_DRIVERSTATUS	Board status (see Common Status Information section)
1	ERHSC_x_y_STATE	Input channel state. BOOL Offset 1 Bit 0 is for channel 1, Bit 1 is channel 2, etc. Each bit will reflect the state, either on or off, of the signal on its respective channel.
4	ERHSC_x_TIMESTAMP	Timestamp of last sample from HSC. This is the number of milliseconds since boot.
8	ERHSC_x_y_COUNTER	Number of counts since boot (Channel 1)
12,16,20, ..., 52	ERHSC_x_y_COUNTER	Counts for Channel 2, 3, 4, ..., 12

Output Map: Max Size: 12 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERHSC_x_y_RESET_COUNT	Reset point counts. BOOL Offset 0, Bit 0 is point 1 ... Offset 1, bit 3 is point 12. Setting a bit to TRUE (1) will reset the count for the point selected. The driver will reset the bit after the count has been reset.
4	ERHSC_x_y_FILTER	30ms/1ms filter select. Offset 4, Bit 0 represent Channel 1 to select 30ms (FALSE) and 1ms (TRUE) respectively. Offset 4, Bit 1 is Channel 2, Offset 5, Bit 3 is Channel 12.
8	ERHSC_x_y_HSC_SEL	High Speed Counter select. Offset 8, Bit 0 represent Channel 1 to select High Speed Counter (TRUE). Offset 8, Bit 1 is Channel 2, Offset 9, Bit 3 is Channel 12. Setting an HSC select bit to TRUE will override the 30ms/1ms selection for the same channel.

ER_STAT ControlWave I/O Expansion Rack Statistics Board

Note:

This is a virtual board; not a physical board in the controller.

DRIVER_NAME: ER_STAT
 DATA_TYPE: BYTE
 DRIVER_PAR1: N/A
 DRIVER_PAR2: IP address - first half
 DRIVER_PAR3: IP address - second half
 DRIVER_PAR4: Specifies redundant board (0 = False, 1 = True)

Input Map: 728 bytes (from RIO)

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0-1	ERSTAT_x_BOARDSTATUS	Board status. Bit 0 - board is not present, Bit 1 - Message parse error, Bit 2 - Memory allocation error, Bit 3 - Board configuration error, Bit 4 - Communications error, Bit 7 - IP channel open in progress.
2-3		Spare
4 (bit 0)	ERSTAT_x_BATSTAT	Expanded IO Status. Battery OK
4 (bit 1)	ERSTAT_x_HOTCARDSTAT	Expanded IO Status. Hot card in progress,
4 (bit 2)	ERSTAT_x_MASTER_IS_B	Expanded IO Status- Is RIO B master?

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
4 (bit 3)	ERSTAT_x_STBYVALID	Expanded IO Status- Is the Standby RIO valid?
4 (bit 4)	ERSTAT_x_FAILOVERERR	Expanded IO Status- Fail-over error
5		Spare
6 (bit 0)	ERSTAT_x_RDN_IO_1_ERR	Failure, readback error, or mismatch with I/O board in redundant standby unit.
6 (bit 1)	ERSTAT_x_RDN_IO_2_ERR	Failure, readback error, or mismatch with I/O board in redundant standby unit.
6 (bit 2)	ERSTAT_x_RDN_IO_3_ERR	Failure, readback error, or mismatch with I/O board in redundant standby unit.
6 (bit 3)	ERSTAT_x_RDN_IO_4_ERR	Failure, readback error, or mismatch with I/O board in redundant standby unit.
6 (bit 4)	ERSTAT_x_RDN_IO_5_ERR	Failure, readback error, or mismatch with I/O board in redundant standby unit.
6 (bit 5)	ERSTAT_x_RDN_IO_6_ERR	Failure, readback error, or mismatch with I/O board in redundant standby unit.
6 (bit 6)	ERSTAT_x_RDN_IO_7_ERR	Failure, readback error, or mismatch with I/O board in redundant standby unit.
6 (bit 7)	ERSTAT_x_RDN_IO_8_ERR	Failure, readback error, or mismatch with I/O board in redundant standby unit.
8-11	ERSTAT_x_HOTCARDCT	Hot card count
12-15	ERSTAT_x_DOWNTIMEUSER	32-Bit Count – Number of seconds (as configured by the user) that the I/O Expansion Rack can be powered off before the outputs are reset to defaults when the unit is powered back up.
16-19	ERSTAT_x_DOWNTIMEACT	32-Bit Count – Number of seconds that the I/O Expansion Rack was powered off on the last power fail.
20-23	ERSTAT_x_WRITECT	Number of messages written to Expanded IO.
24-27	ERSTAT_x_READCT	Number of update message sent to host.
28-31	ERSTAT_x_CONNECTS	Number of times the Expanded IO has connected to a host (open + close).
32-35	ERSTAT_x_HEARTBEAT	Number of heartbeat messages sent to a host.
36-41		Spare
42-126	ERSTAT_x_BDSTR1	Expanded IO local board string information for slot #1.
127-211	ERSTAT_x_BDSTR2	Expanded IO local board string information for slot #2.
212-296	ERSTAT_x_BDSTR3	Expanded IO local board string information for slot #3.
297-381	ERSTAT_x_BDSTR4	Expanded IO local board string information for slot #4.
382-466	ERSTAT_x_BDSTR5	Expanded IO local board string information for slot #5.
467-551	ERSTAT_x_BDSTR6	Expanded IO local board string information for slot #6.
552-636	ERSTAT_x_BDSTR7	Expanded IO local board string information for slot

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
		#7.
637-722	ERSTAT_x_BDSTR8	Expanded IO local board string information for slot #8.
724-727	ERSTAT_x_REDUNSTAT	Redundancy status
728-812	ERSTAT_x_BDSTR9	Reserved for possible future use.
813-897	ERSTAT_x_BDSTR10	Reserved for possible future use.
898-982	ERSTAT_x_BDSTR11	Reserved for possible future use.
983-1067	ERSTAT_x_BDSTR12	Reserved for possible future use.
1068-1152	ERSTAT_x_BDSTR13	Reserved for possible future use.
1153-1237	ERSTAT_x_BDSTR14	Reserved for possible future use.
1240	ERSTAT_x_INPUTVOLTS	Float – Reading (in volts) of power-supply input voltage.

Output Map: 4 bytes (to RIO)

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERSTAT_x_FAILOVER_O	Bit 0 - Force redundant Expanded IO to become standby unit.
1	ERSTAT_RDN_IOERR_WARN	When set TRUE, I/O errors are treated only as warnings. When FALSE, I/O errors disable redundancy.
2-3		Spare

ER_TC12 ControlWave I/O Expansion Rack– 12 Point Thermocouple Board

DRIVER_NAME	'ER_TC12'
DATA_TYPE	DWORD (32 bits)
DRIVER_PAR1	slot number.
DRIVER_PAR2	First two bytes of Primary IP address
DRIVER_PAR3	Lower two bytes of Primary IP address
DRIVER_PAR4	If Bit 0 is 1, this is a redundant Expansion Rack
Input Map:	Max Size: 56 bytes

Due to the amount of time required to process the thermocouple points, it is highly recommended that this I/O driver be assigned to a task (instead of "No Task"). Care should also be taken in using the I/O board in a task of less than 40ms.

Offset	Default Variable Name where <i>x</i> is the board slot and <i>y</i> is the pin number.	Description
0	ERTC_x_DRIVERSTATUS	Board Status. See Common Status Information section.
2 (Bit 2)	ERTC_x_CALIBRATE	If set indicates invalid calibration data written to the board.
2 (Bit 3)	ERTC_x_TIMEOUT	If set, indicates that had an error reading or writing to the board.
4	ERTC_x_y_OUTRANGE	1 bit per TC, TC1 is bit 0, TC8 is bit 7. If set, input is Out-of-range.
5	ERTC_x_y_OUTRANGE	1 bit per TC, TC9 is bit 0, TC12 is bit 3. If set, input is Out-of-range.
8	ERTC_x_y	Value for TC1 in engineering units (4-byte float - REAL). To access the value, define the variable AT %IDxx. Direct access to %IDxx is not possible.
12, 16, 20, 24, ... 52	ERTC_x_y	Value for TC2, TC3, TC4, TC5, TC6... TC12

Output Map: Max Size: 112 bytes

Offset	Default Variable Name where <i>x</i> is the board slot and <i>y</i> is the pin number.	Description
0	ERTC_x_y_ZERO	Zero for TC1 (4-byte float - REAL). To access the value, define the variable AT %QDxx. This variable can be initialized at declaration.
4	ERTC_x_y_SPAN	Span for TC1 (4-byte float). If zero, the TC will be scaled as in the chart below. If specified, the new value will be $ORG_VALUE * Span + Zero$.
8, 16, 24, 32, ... 88	ERTC_x_y_ZERO	Zeros for TC2, TC3, TC4, to TC12 – Example: for C to F, use 32.0
12, 20, 28, 36, ... 92	ERTC_x_y_ZERO	Spans for TC2, TC3, TC4, to TC12 – Example for C to F, use 1.8
100	ERTC_x_y_MODE	Point type for TC1; see Thermocouple type codes section for details.
101, 102, 103, 104, ... 111	ERTC_x_y_MODE	Point types for TC2, TC3, TC4, to TC12.

Type codes for Thermocouple Points.

Type Code	Code	Range
0	B	Thermocouple: 100C – +1820C
1	E	Thermocouple: -270C – +1000C
2	J	Thermocouple: -210C – +1200C
3	K	Thermocouple: -270C – +1370C
4	R	Thermocouple: -50C – +1720C
5	S	Thermocouple: -50C – +1760C

Type Code	Code	Range
6	T	Thermocouple: -270C – +400C
7	Unused	Unused
8	10MV	Voltage Inputs: -10 mV to +10 mV (Outputs as 0.0 to 1.0)
9	C	Thermocouple: 0C – +2315C
10	N	Thermocouple: -270C – +1300C

ER_RTD8 - ControlWave I/O Expansion Rack – 8 Point Resistance Temperature Device (RTD) Board

DRIVER_NAME	'ER_RTD8'
DATA_TYPE	DWORD (32 bits)
DRIVER_PAR1	slot number.
DRIVER_PAR2	First two bytes of Primary IP address
DRIVER_PAR3	Lower two bytes of Primary IP address
DRIVER_PAR4	If Bit 0 is 1, this is a redundant Expansion Rack
Input Map:	Max Size: 40 bytes

Due to the amount of time required to process the RTD points, it is highly recommended that this I/O driver be assigned to a task (instead of “No Task”). Care should also be taken in using the I/O board in a task of less than 40ms.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERRTD_x_DRIVERSTATUS	Board Status. See Common Status Information section.
2 (Bit 2)	ERRTD_x_CALIBRATE	If set indicates invalid calibration data written to the board.
2 (Bit 3)	ERRTD_x_TIMEOUT	If set, indicates that had an error reading or writing to the board.
3 (Bit 0)	ERRTD_x_LASTCALBOP	Set if last calibration or reset operation failed.
3 (Bit 7)	ERRTD_x_CALBCMD	Calibration Commands Allowed. Until this bit is set, all calibration commands are ignored.
4	ERRTD_x_y_READERR	RTD Reading Error. Bit 0 is RTD1, Bit 7 is RTD8
8	ERRTD_x_y	RTD1 reading – REAL – In units of Degrees Centigrade (unless scaled by values in the output map).
12, 16, ...36	ERRTD_x_y	Readings for RTD2 ... RTD8.

Output Map: Max Size: 280 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERRTD_x_y_ZERO	Zero for RTD 1(4-byte float - REAL). Example: for

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
		C to F, use 32.0. Defaults to 0.0
4	ERRTD_x_y_SPAN	Span for RTD 1(4-byte float – REAL). If zero, RTD will not be scaled. If specified, the scaled value will be $ORG_VALUE * Span + Zero$. Example for C to F, use 1.8.
8,16,24,32 40,48,56	ERRTD_x_y_ZERO	Zero for RTDs 2-8
12,20,28, 36,44,52, 60	ERRTD_x_y_SPAN	Span for RTDs 2-8
64		NOT USED; RESERVED FOR FUTURE USE
65,66,67, 68,69		NOT USED; RESERVED FOR FUTURE USE
100	ERRTD_x_y_MODE	RTD 1 Type
101,102 103,104 105,106, 107	ERRTD_x_y_MODE	RTDs 2-8 Type
120 (Bit 0) *	ERRTD_x_y_RESTORE	If set, restore RTD 1 calibration to Factory Defaults. Will be reset when operation completes.
121 *	ERRTD_x_y_OPERATION	Calibration Operation for RTD 1 – SINT 5 RTD Zero (100 Ohms) 6 RTD Span (300 Ohms) 7 RTD Coefficients (A, B, R0) 8 RTD Span (not using 300 Ohms)
124 **	ERRTD_x_y_COEFF_A	Coefficient A (RTD 1)
128 **	ERRTD_x_y_COEFF_B	Coefficient B (RTD 1)
132 **	ERRTD_x_y_COEFF_R0	Coefficient R0 (RTD 1)
136 **	ERRTD_x_y_APPLIED	The applied temperature when calibration operation 8 was performed. (RTD 1)
140 (Bit 0) *	ERRTD_x_y_RESTORE	If set, restore RTD 2 calibration to Factory Defaults. Will be reset when operation completes.
141 *	ERRTD_x_y_OPERATION	Calibration Operation for RTD 2 – SINT 5 RTD Zero (100 Ohms) 6 RTD Span (300 Ohms) 7 RTD Coefficients (A, B, R0) 8 RTD Span (not using 300 Ohms)
144*	ERRTD_x_y_COEFF_A	Coefficient A (RTD 2)
148**	ERRTD_x_y_COEFF_B	Coefficient B (RTD 2)
152*	ERRTD_x_y_COEFF_R0	Coefficient R0 (RTD 2)
156**	ERRTD_x_y_APPLIED	The applied temperature when calibration operation 8 was performed. (RTD 2)
....	
260 (Bit 0) *	ERRTD_x_y_RESTORE	If set, restore RTD 8 calibration to Factory

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
		Defaults. Will be reset when operation completes.
261 *	ERRTD_x_y_OPERATION	Calibration Operation for RTD 8 – SINT 5 RTD Zero (100 Ohms) 6 RTD Span (300 Ohms) 7 RTD Coefficients (A, B, R0) 8 RTD Span (not using 300 Ohms)
264 **	ERRTD_x_y_COEFF_A	Coefficient A (RTD 8)
268 **	ERRTD_x_y_COEFF_B	Coefficient B (RTD 8)
272 **	ERRTD_x_y_COEFF_R0	Coefficient R0 (RTD 8)
276 **	ERRTD_x_y_APPLIED	The applied temperature when calibration operation 8 was performed. (RTD 8)

* Value written to perform operation. The value will be reset by driver when the operation completes.

** Value is read from the board by the driver. In order to perform calibration operations 7 and 8, the user can overwrite the values; then, issue the calibration command.

Local I/O – ControlWave MICRO-series

The following sections describe the local I/O boards supported, and their memory maps.

Offset 0 of the input map defines the board status. All boards support bit 0, which is set if the board is not present.

CWM_DO16 – ControlWave MICRO 16 pin Digital Output Board

DRIVER_NAME 'CWM_DO16'

DATA_TYPE BYTE

DRIVER_PAR1 slot number.

Input Map: Max Size: 6 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	DO16_x_BOARDSTATUS	Board status.
4,5	DO16_x_y_I	DO status as seen by card. 1 bit per value.

Output Map: Size: 5 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	DO16_x_y	Outputs. 1 bit per value. DO1 is LSB; DO8 is MSB.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
		Typically specified as %QXy.z, where y is I/O space offset, and z is bit number from 0 to 7.
1	DO16_x_y	Outputs 9 to 16.
4	DO16_x_LEDSTATUS	Single bit. If bit is set, the diagnostic LEDs for the points are turned off to save power.

CWM_DI16 – ControlWave MICRO 16 pin Digital Input Board

DRIVER_NAME 'CWM_DI16'
 DATA_TYPE DWORD (32 bits)
 DRIVER_PAR1 s lot number.
 Input Map: Max Size: 8 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number	Description
0	DI16_x_BOARDSTATUS	Board status.
4	DI16_x_y	Current status of DI1 (in bit 0) to DI8 (in bit 7).
5	DI16_x_y	Current status of DI9 (in bit 0) to DI16 (in bit 7).

Output Map: Size: 5 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number	Description
4	DI16_x_LEDSTATUS	Single bit. If bit is set, the diagnostic LEDs for the points are turned off to save power.

CWM_MD – ControlWave MICRO - Mixed Digital Board 12 Digital Input Pins / 4 Digital Output Pins

DRIVER_NAME 'CWM_MD'
 DATA_TYPE BYTE
 DRIVER_PAR1 slot number.
 Input Map: Max Size: 9 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number	Description
0	DIDO_x_BOARDSTATUS	Board status.
4	DIDO_x_y	Current status of DI1 (in bit 0) to DI8 (in bit 7).
5	DIDO_x_y	Current status of DI9 (in bit 0) to DI12 (in bit 3).
8	DIDO_x_y_O_I	DO status as seen by card. 1 bit per value.

Output Map: Size: 5 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number	Description
0	DIDO_x_y_O	Outputs. 1 bit per value. DO1 is LSB; DO4 is MSB. Typically specified as %QXy.z, where y is I/O space offset, and z is bit number from 0 to 3.
4	DIDO_x_LEDSTATUS	Single bit. If bit is set, the diagnostic LEDs for the points are turned off to save power.

CWM_AI8 – ControlWave MICRO 8 Analog Input Pin Board

DRIVER_NAME 'CWM_AI8'
 DATA_TYPE DWORD (32 bits)
 DRIVER_PAR1 slot number.
 Input Map: Max Size: 40 bytes

Due to the amount of time required to process the AI points, it is highly recommended that the input region for this board be sized only as large as needed for the points used by the application.

Also, the I/O fetches should be programmed to only occur as fast as needed.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0 (bit 0)	AI8_x_y_BOARDSTATUS	Board status. Bit 0 is set if board is not present.
0 (bit 2)	AI8_x_CALIBRATE	Calibration error. Bit set on error. Indicates a serious hardware failure on the board.
0 (bit 3)	AI8_x_TIMEOUT	Timeout – If board timeout occurs, bit set. Indicates a serious hardware failure on the board.
4	AI8_x_y_OUTRANGE	1 bit per AI, AI1 is bit 0, AI8 is bit 7. If set, input is Out-of-range.
8	AI8_x_y	Value for AI1 in engineering units (4-byte float – REAL). To access the value, define the variable AT %IDxx. Direct access to %IDxx is not possible.
12, 16, ...	AI8_x_y	Value for AI2, AI3, ...

Output Map: Max Size: 64 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	AI8_x_y_ZERO	Zero for AI1 (4-byte float - REAL). To access the value, define the variable AT %QDxx. This variable can be initialized at declaration.
4	AI8_x_y_SPAN	Span for AI1 (4-byte float). If zero, the AI will be scaled from 0 to 100.0.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
8, 16, 24, ...	AI8_x_y_ZERO	Zeros for AI2, AI3, ...
12, 20, 28, ...	AI8_x_y_SPAN	Spans for AI2, AI3, ...
64, 72, 80, ...	AI8_x_y_BOTTOMRANGE	Bottom end of usable current / voltage range for this input. Specified as a REAL. For example, for a 1-5V input, this value is 1.0.
68, 76, 84, ...	AI8_x_y_TOPRANGE	Top end of usable current / voltage range for this input. Specified as a REAL. For example, for a 1-5V input, this value is 5.0.
128	AI8_x_y_MODE	Mode - 1 bit per AI; set TRUE if the point is voltage; FALSE if current.

CWM_AO4 – ControlWave MICRO 4 Analog Output Pin Board

DRIVER_NAME 'CWM_AO4'

DATA_TYPE DWORD (32 bits)

DRIVER_PAR1 slot number.

Input Map: Max Size: 24 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0 (bit 0)	AO4_x_BOARDSTATUS	Board status. Bit 0 is set if board is not present; Bit 1 is set if the last conversion operation failed.
0 (bit 2)	AO4_x_CALIBRATE	Calibration error. Bit set on error. Indicates a serious hardware failure on the board.
0 (bit 3)	AO4_x_TIMEOUT	Timeout – If board timeout occurs, bit set. Indicates a serious hardware failure on the board.
6	AO4_x_y_OUTRANGE	1 bit per AO, AO0 is bit 0, AO4 is bit 3. If set, output is Out-of-range.
8,12,16,20	AO4_x_y_ACTUAL	Real value of value actually output. Clamped to 0-100% of scale.

Output Map: Max Size: 48 bytes

Due to the amount of time required to process the AO points, it is highly recommended that the output region for this board be sized only as large as needed for the points used by the application.

Also, the I/O sets could be programmed to only occur as fast as needed.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	AO4_x_y_ZERO	Zero for AO1 (4-byte float - REAL). To access the value,

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
		define the variable <i>AT %QDxx</i> . This variable can be initialized at declaration.
4	AO4_x_y_SPAN	Span for AO1 (4-byte float). If zero, the AO will be scaled from 0 to 100.0.
8	AO4_x_y	Value for AO1 (4-byte float).
12, 24, 36	AO4_x_y_ZERO	Zeros for AO2, AO3, AO4
16, 28, 40	AO4_x_y_SPAN	Spans for AO2, AO3, AO4
20, 32, 44	AO4_x_y	Values for AO2, AO3, AO4

CWM_MA – ControlWave MICRO - Mixed Analog Board - 6 Analog Input Pin / 2 Analog Output Pin Board

DRIVER_NAME 'CWM_MA'

DATA_TYPE DWORD (32 bits)

DRIVER_PAR1 slot number.

Input Map: Max Size: 40 bytes

Due to the amount of time required to process the AI points, it is highly recommended that the input region for this board be sized only as large as needed for the points used by the application.

Also, the I/O fetches should be programmed to only occur as fast as needed.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0 (bit 0)	AIAO_x_BOARDSTATUS	Board status. Bit 0 is set if board is not present (or if either Bit 2 or Bit 3 is set).
0 (bit 2)	AIAO_x_CALIBRATE	Calibration error. Bit set on error. Indicates a serious hardware failure on the board.
0 (bit 3)	AIAO_x_TIMEOUT	Bit 3 is set to indicate that a board data read has timed out. Indicates a serious hardware failure on the board.
4	AIAO_x_y_OUTRANGE	1 bit per AI, AI1 is bit 0, AI6 is bit 5. If set, input is Out-of-range.
6	AIAO_x_y_O_OUTRANGE	1 bit per AO, AO0 is bit 0, AO2 is bit 1. If set, output is Out-of-range.
8	AIAO_x_y	Value for AI1 in engineering units (4-byte float - REAL). To access the value, define the variable <i>AT %IDxx</i> . Direct access to <i>%IDxx</i> is not possible.
12, 16, ...	AIAO_x_y	Value for AI2, AI3, ...
32,36	AIAO_x_y_O_ACTUAL	Value actually written to AO – clamped to 0-100%.

Output Map: Max Size: 72 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	AIAO_x_y_ZERO	Zero for AI1 (4-byte float - REAL). To access the value, define the variable AT %QDxx. This variable can be initialized at declaration.
4	AIAO_x_y_SPAN	Span for AI1 (4-byte float). If zero, the AI will be scaled from 0 to 100.0.
8, 16, 24, ...	AIAO_x_y_ZERO	Zeros for AI2, AI3, ...
12, 20, 28, ...	AIAO_x_y_SPAN	Spans for AI2, AI3, ...
48, 60	AIAO_x_y_O_ZERO	Zeros for AO1, AO2
52, 64	AIAO_x_y_O_SPAN	Spans for AO1, AO2
56, 68	AIAO_x_y_O	Values for AO1, AO2

CWM_AI6 – ControlWave MICRO - 6 Analog Input Pin Board

DRIVER_NAME 'CWM_AI6'
 DATA_TYPE DWORD (32 bits)
 DRIVER_PAR1 slot number.

Input Map: Max Size: 40 bytes

Due to the amount of time required to process the AI points, it is highly recommended that the input region for this board be sized only as large as needed for the points used by the application.

Also, the I/O fetches should be programmed to only occur as fast as needed.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0 (bit 0)	AI6_x_BOARDSTATUS	Board status. Bit 0 is set if board is not present (or if either Bit 2 or Bit 3 is set);
0 (bit 2)	AI6_x_CALIBRATE	Calibration error. Bit set on error. Indicates a serious hardware failure on the board.
0 (bit 3)	AI6_x_TIMEOUT	Bit 3 is set to indicate that a board data read has timed out. Indicates a serious hardware failure on the board.
4	AI6_x_y_OUTRANGE	1 bit per AI, AI1 is bit 0, AI6 is bit 5. If set, input is Out-of-range.
6		Unused
8	AI6_x_y	Value for AI1 in engineering units (4-byte float - REAL). To access the value, define the variable AT %IDxx. Direct access to %IDxx is not possible.
12, 16, ...	AI6_x_y	Value for AI2, AI3, ...
32 to 36		Unused

Output Map: Max Size: 72 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	AI6_x_y_ZERO	Zero for AI1 (4-byte float - REAL). To access the value, define the variable AT %QDxx. This variable can be initialized at declaration.
4	AI6_x_y_SPAN	Span for AI1 (4-byte float). If zero, the AI will be scaled from 0 to 100.0.
8, 16, 24, ...	AI6_x_y_ZERO	Zeros for AI2, AI3, ...
12, 20, 28, ...	AI6_x_y_SPAN	Spans for AI2, AI3, ...
48 to 68		Unused

CWM_HSC4 – ControlWave MICRO - 4 channel High Speed Counter Board

DRIVER_NAME 'CWM_HSC4'

DATA_TYPE DWORD (32 bits)

DRIVER_PAR1 slot number.

Input Map: Max Size: 24 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	HSC4_x_BOARDSTATUS	Board status. Only bit 0 is currently defined. If set, board is not present.
4	HSC4_x_TIMESTAMP	Timestamp of last sample from HSC. This is the number of milliseconds since boot.
8	HSC4_x_y_COUNTER	Number of counts since boot (Channel 1)
12, 16, 20	HSC4_x_y_COUNTER	Counts for Channel 2, 3, 4

Output Map: Size: 8 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	HSC4_x_y_RESET_COUNT	Counter reset flags, HSC1 (in bit 0) to HSC4 (in bit 3)
2	HSC4_x_NOINIT	Bit 1 – If set to TRUE, maintain counts across warm start.
4	HSC4_x_LEDSTATUS	Single bit. If bit is set, the diagnostic LEDS for the

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
		points are turned off to save power.

CWM_BAT – ControlWave MICRO Battery (Voltage) Monitor

DRIVER_NAME 'CWM_BAT'
 DATA_TYPE DWORD (32 bits)
 DRIVER_PAR1 slot number should be specified as 0.
 Input Map: Max Size: 8 bytes

Due to the expense of performing the A-to-D conversion for this value, this input should be fetched only when needed.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	BAT_x_STATUS	Board status. Board is always present, therefore is always 0.
4	BAT_x_READING	Value for Input Voltage in engineering units (4-byte float – REAL – 0 to 32V range). To access the value, define the variable AT %IDxx. Direct access to %IDxx is not possible.

CWM_MIX – ControlWave MICRO - 6 DI/O, 4AI, 1AO (optional), 2 HSC - Mixed I/O Board

This card contains the following I/O: DIO 6, AI4, AO1 (optional), and HSC2

DRIVER_NAME 'CWM_MIX'
 DATA_TYPE DWORD
 DRIVER_PAR1 slot number.
 Input Map: Max Size: 68 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0 (bit 0)	MIX_x_BOARDSTATUS	Board status.
0 (bit 2)	MIX_x_CALIBRATE	Calibration error. Bit set on error. Indicates a serious hardware failure on the board.
0 (bit 3)	MIX_x_TIMEOUT	Timeout – If board timeout occurs, bit set. Indicates a serious hardware failure on the board.
4	MIX_x_y_AI_OUTRANGE	AI under / over range. One bit per point: AI1 in bit 0, AI4 in bit 3.
6	MIX_x_y_AO_OUTRANGE	AO under / over range. AO1 is in bit 0.
8	MIX_x_y_DI	Current status of DI1 (in bit 0) to DI6 (in bit 5).
10	MIX_x_y_DO_I	Read-back of current value for DO1 (in bit 0) to DO6

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
		(in bit 5).
12	MIX_x_TIMESTAMP	Timestamp of last sample from HSC. This is the number of milliseconds since boot.
16	MIX_x_y_COUNTER	Number of counts since boot (Channel 1)
20	MIX_x_y_COUNTER	Counts for Channel 2
32	MIX_x_y_AI	Value for AI1 in engineering units (4-byte float - REAL). To access the value, define the variable AT %IDxx. Direct access to %IDxx is not possible.
36, 40, ...	MIX_x_y_AI	Value for AI2, AI3, ...
64	MIX_x_y_AO_ACTUAL	Value actually written to AO – clamped to 0-100%.

Output Map:

Size: 84 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	MIX_x_y_DO	Outputs. 1 bit per value. DO1 is LSB; DO6 is MSB. Typically specified as %QXy.z, where y is I/O space offset, and z is bit number from 0 to 5.
2	MIX_x_y_RESET_COUNT	Counter reset flags, HSC1 (in bit 0) to HSC2 (in bit 1)
4 (Bit 0)	MIX_x_LEDSTATUS	Bit 0 - If bit is set, the diagnostic LEDs for the points are turned off to save power.
4 (Bit 1)	MIX_x_NOINIT	Bit 1 – If set to TRUE, maintain counts across warm start.
8	MIX_x_y_AI_ZERO	Zero for AI1 (4-byte float - REAL). To access the value, define the variable AT %QDxx. This variable can be initialized at declaration.
12	MIX_x_y_AI_SPAN	Span for AI1 (4-byte float). If zero, the AI will be scaled from 0 to 100.0.
16, 24, 32	MIX_x_y_AI_ZERO	Zeros for AI2, AI3, AI4
20, 28, 36	MIX_x_y_AI_SPAN	Spans for AI2, AI3, AI4
72	MIX_x_y_AO_ZERO	Zero for AO1 (4-byte float - REAL). To access the value, define the variable AT %QDxx. This variable can be initialized at declaration.
76	MIX_x_y_AO_SPAN	Span for AO1 (4-byte float). If zero, the AO will be scaled from 0 to 100.0.
80	MIX_x_y_AO	Value for AO1 (4-byte float).

CWM_SCB – ControlWave EFM System Controller Board

Firmware release 04.20 supports the battery (incoming) voltage reading only. Unless otherwise noted, firmware release 04.30 and newer supports all of the items shown, below.

DRIVER_NAME 'CWM_SCB'
 DATA_TYPE DWORD (32 bits)
 DRIVER_PAR1 slot number should be specified as 0.
 Input Map: Max Size: 40 bytes

Offset	Default Variable Name where x = board slot number	Description
0 (Bit 0)	SCB_x_BOARDSTATUS	Board status. If SCB is present, will be FALSE; if Micro Power Supply is present, it will be TRUE.
0 (Bit 1)	SCB_x_PSRDERR	Pressure Reading Error. If set, pressure reading is not valid.
0 (Bit 2)	SCB_x_PSRDINITERR	Pressure Reading – Initialization failure – no retry performed on this error.
0 (Bit 3)	SCB_x_PSRDCOMMERR	Pressure Reading – Communications failure – will be retried.
0 (Bit 4)	SCB_x_PSRDNORESET	Pressure Reading – Wet End has been changed without power reset.
0 (Bit 5)	SCB_x_RTRDERR	RTD Reading Error.
0 (Bit 6)	SCB_x_BATRDERR	Battery Voltage Reading Error.
0 (Bit 7)	SCB_x_ALLOWCALIB	Calibration Commands Allowed. Until this bit is set, all calibration commands are ignored.
1 (Bit 0)	SCB_x_LASTCALBOP	Set if last calibration or reset operation failed.
3	SCB_x_BOARDPRESENT	I/O board present. See Board Type Table code numbers, later in this section.
4	SCB_x_INVOLT	Value for Input Voltage in engineering units (4-byte float – REAL – 0 to 32V range).
8	SCB_x_DP	Differential Pressure – REAL – In units of Inches of Water (inH2O) or PSI. The units used depend on the range type of the connected transmitter (see chart below).
12	SCB_x_SP	Static Pressure – REAL – In units of Pounds per Square Inch (PSI).
16	SCB_x_RTD	RTD reading – REAL – In units of Degrees Centigrade.
20	SCB_x_TEMP	Estimated Sensor Temperature – REAL – In units of Degrees Centigrade.
24	SCB_x_SENSORID	Sensor ID (Block Number) –INT
26	SCB_x_SENORSERIES	Sensor Series - SINT
28	SCB_x_SENSORTYPE	Sensor Type – SINT
29	SCB_x_DPRANGE	DP Range Code – SINT
30	SCB_x_SPRANGE	SP Range Code – SINT
31 (Bit 0)	SCB_x_DPENABLE	Differential Pressure Processing is enabled.
32 (Bit 0)	SCB_x_SPENABLE	Static Pressure Processing is enabled.
33 (Bit 0)	SCB_x_RTDENABLE	RTD Processing is enabled.
36	SCB_x_LASTOPERR	DWORD – Specific Error for last operation – See codes below.

Offset	Default Variable Name where x = board slot number	Description
40	SCB_x_MSPTMP	Not applicable; not used for this platform.

Transmitter types:

Sensor Type	DP Range Code	Range / Units
32 (Differential Pressure)	12	150 InH2O
32 (Differential Pressure)	13	100 InH2O
32 (Differential Pressure)	14	300 InH2O
32 (Differential Pressure)	20	25 psi
12 (Gauge Pressure)	14	300 InH2O
12 (Gauge Pressure)	20	25 psi
12 (Gauge Pressure)	22	100 psi
12 (Gauge Pressure)	23	300 psi
12 (Gauge Pressure)	25	1000 psi
12 (Gauge Pressure)	26	3000 psi
12 (Gauge Pressure)	28	2000 psi
12 (Gauge Pressure)	29	4000 psi

Note: If DP Range Code is < 20, units are InH2O, if >= 20, units are psi.

Sensor Type	SP Range Code	Range / Units
32 (Differential Pressure)	1	1000 psi
32 (Differential Pressure)	2	2000 psi
32 (Differential Pressure)	3	500 psi
32 (Differential Pressure)	4	4000 psi

Specific Error Codes (this is an addition of the following bit values:

- 0x00000001 Could not erase MSP Info memory
- 0x00000002 Could not write MSP Info memory
- 0x00000004 Checksum failure in Serial EEPROM.
- 0x00000008 Flash checksum failure in Sensor.
- 0x00000010 Flash checksum failure in MSP Info.
- 0x00000100 DP - Could not initialize the SA.
- 0x00000200 DP - SA flash checksum error.
- 0x00000400 DP - Data error after Init done.
- 0x00000800 DP - Data error during scan.
- 0x00001000 DP - Data scan error.
- 0x00010000 SP - Could not initialize the SA.
- 0x00020000 SP - SA flash checksum error.

0x00040000 SP - Data error after Init done.

0x00080000 SP - Data error during scan.

0x00100000 SP - Data scan error.

0x01000000 Could not initialize the ADS.

0x02000000 Data scan error.

0x04000000 RTD out of range.

0x08000000 Temperature calculation error.

Output Map: Max Size: 28 bytes

Offset	Default Variable Name where x = board slot number	Description
0 (Bit 0)*	SCB_x_RESTOREDP	Restore Factory Calibration Values for Differential Pressure.
1 (Bit 0)*	SCB_x_RESTORESP	Restore Factory Calibration Values for Static Pressure.
2 (Bit 0)*	SCB_x_RESTORERTD	Restore Factory Values for RTD.
3 **	SCB_x_CALIBOP	Perform Calibration Operation – SINT 1 DP Zero 2 DP Span 3 SP Zero 4 SP Span 5 RTD Zero (100 Ohms) 6 RTD Span (300 Ohms) 7 RTD Coefficients (A, B, R0) 8 RTD Span (not using 300 Ohms)
8 **	SCB_x_DPSPAN	DP Applied at DP Span Calibration – REAL – Units of Inches of Water..
12 **	SCB_x_SPSPAN	SP Applied at SP Span Calibration – REAL – Units of Pounds per Square Inch.
16 ***	SCB_x_RTDA	RTD Coeff – A
20 ***	SCB_x_RTDB	RTD Coeff – B
24 ***	SCB_x_RTDR0	RTD Coeff – R0
28	SCB_x_RTDSpan	The applied temperature when calibration operation 8 was performed (RTD 8).

* Value is cleared by the driver when the operation completes.

** Value is read from the hardware at startup and after any calibration operation. User changes to this value only can be made when Bit 7 of Offset 0 in the input map is set.

*** Same as **; but, reserved for future expansion.

Note:

For any REAL variables, a variable needs to be defined at the proper address; direct access to the memory location via %IDxx is not possible.

CWM_HIB – ControlWave MICRO – HART Interface Board

DRIVER_NAME 'CWM_HIB'
 DATA_TYPE DWORD (32 bits)
 DRIVER_PAR1 Slot number
 Input Map: Max Size: 164 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description								
0 (Bit 0)	HIB_x_BOARDSTATUS	Board status. Bit 0 is set if board is not present.								
0 (Bit 2)	HIB_x_CALIBRATE	Calibration Error. AI and/or AO calibration data is invalid.								
0 (Bit 3)	HIB_x_TIMEOUT	Timeout – If board timeout occurs, bit set. Indicates a serious hardware failure on the board.								
4, 5	HIB_x_y_OUTRANGE	1 bit per Point. If set, input or output is Out-of-range.								
8	HIB_x_y_I	Value for AI1 in engineering units (4-byte float - REAL).								
12, 16, ...68	HIB_x_y_I	Value for points 2 to 16								
72..87	HIB_x_y_STATUS	USINT per point: 00 = no error 01 = switch set incorrectly for I/O configuration. Switch errors result from any of the following three error cases: <table border="1" data-bbox="776 1108 1383 1352"> <thead> <tr> <th>HART channel configured in software (I/O Configurator) as:</th> <th>Switch SW3 (channel 1) or Switch SW4 (channel 2) incorrectly set to:</th> </tr> </thead> <tbody> <tr> <td>Analog Input</td> <td>Output</td> </tr> <tr> <td>Analog Output</td> <td>Input</td> </tr> <tr> <td>HART Multidrop</td> <td>Channel(s) set to Output</td> </tr> </tbody> </table> <p>Note: For multi-drop, both SW3 and SW4 must be set as inputs. Neither can be outputs.</p>	HART channel configured in software (I/O Configurator) as:	Switch SW3 (channel 1) or Switch SW4 (channel 2) incorrectly set to:	Analog Input	Output	Analog Output	Input	HART Multidrop	Channel(s) set to Output
HART channel configured in software (I/O Configurator) as:	Switch SW3 (channel 1) or Switch SW4 (channel 2) incorrectly set to:									
Analog Input	Output									
Analog Output	Input									
HART Multidrop	Channel(s) set to Output									
88-99		Padding – Align AO actual at offset = 100								
100	HIB_x_y_ACTUAL	AO - Actual value written to board. (4-byte float - REAL).								
104, 108, .. 160	HIB_x_y_ACTUAL	AO - Actual value for points 2 to 16								

Output Map: Size:208 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	HIB_x_y_ZERO	Zero for Point 1 (4-byte float – REAL).
4	HIB_x_y_SPAN	Span for Point 1 (4-byte float – REAL)
8..120	HIB_x_y_ZERO	Zeroes for Points 2 – 16
12..124	HIB_x_y_SPAN	Spans for Points 2 – 16

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
128	HIB_x_y_O	AO value to be written for point 1. Ignored for non-AO point types.
132..188	HIB_x_y_O	AO value for points 2 – 16
192..207	HIB_x_y_TYPE	USINT per point. Indicates how the point is to be configured: 0 = Unconfigured 1 = AI 2 = AO 3 = HART Multidrop 4 = BBTI. NOTE: AI and AO modes allow HART point to point communications. Setting the configuration to a valid non-zero value will lock the channel to that configuration.

Local I/O – ControlWave GFC-CL and ControlWave XFC

The following sections describe the local I/O boards supported, and their memory maps.

Offset 0 of the input map defines the board status. All boards support bit 0, which is set if the board is not present.

CWM_RTU - Mixed I/O board

This card contains the following I/O: DIO 6 (previously 4), AI3, AO1, HSC2, Battery Voltage, RTD, and Wet-End interface.

Notes:

- Due to the design of the SPI interface connecting the CPU to the I/O board, the I/O can only be read / written at one second intervals.
- The GFC and XFC share this I/O map. Points for which there is no physical hardware will not function (and a status is provided to indicate which I/O sub-system is present).
- Two versions of this board have been produced. The main difference between them is that the original board (P/N 400-075-00) supported 4 DIO, whereas the newer board (P/N 400-132-00) supports 6 DIO.

DRIVER_NAME	'CWM_RTU'
DATA_TYPE	DWORD
DRIVER_PAR1	slot number – specify as 0.
Input Map:	Max Size: 136 bytes

Offset	Default Variable Name where x = board slot number	Description
0 (Bit 0)	MIX_x_BOARDSTATUS	Board status. If this SCB is present, status is FALSE; TRUE indicates that the board is not plugged in.
0 (Bit 2)	MIX_x_CALIBRATE	Calibration Error. AI and/or AO calibration data is invalid.
0 (Bit 3)	MIX_x_TIMEOUT	Timeout occurred communicating with I/O board.
1 (Bit 1)	MIX_x_PSRDERR	Pressure Reading Error. If set, pressure reading is not valid.
1 (Bit 2)	MIX_x_PSRDINITERR	Pressure Reading – Initialization failure – no retry performed on this error.
1 (Bit 3)	MIX_x_PSRDCOMMERR	Pressure Reading – Communications failure – will be retried.
1 (Bit 4)	MIX_x_PSRDNORESET	Pressure Reading – Wet End has been changed without power reset.
1 (Bit 5)	MIX_x_RTDRDERR	RTD Reading Error.
1 (Bit 6)	MIX_x_BATRDERR	Battery Voltage Reading Error.
1 (Bit 7)	MIX_x_ALLOWCALIB	Calibration Commands Allowed. Until this bit is set, all calibration commands are ignored.
2 (Bit 0)	MIX_x_LASTCALBOP	Set if last calibration or reset operation failed.
3	MIX_x_BOARDPRESENT	I/O board present. See Board Type Table code numbers, later in this section.
4	MIX_x_AI_OUTRANGE	AI under / over range. One bit per point: AI1 in bit 0, AI3 in bit 2.
6	MIX_x_y_AO_OUTRANGE	AO under / over range. AO1 is in bit 0.
8	MIX_x_1_DI, MIX_x_2_DI	Current status of DI1 (in bit 0) to DI6 (in bit 5) NOTE: Older boards only had DI1 and DI2 (bits 0 and 1).
10	MIX_x_1_DO_I MIX_x_2_DO_I	Read-back of current value for DO1 (in bit 0) to DO4 (in bit 3). NOTE: Older boards only had DO1 and DO2 (bits 0 and 1).
12	MIX_x_TIMESTAMP	Timestamp of last sample from HSC. This is the number of Milliseconds since boot.
16	MIX_x_1_COUNTER	Number of counts since boot (Channel 1)
20	MIX_x_2_COUNTER	Counts for Channel 2
32	MIX_x_1_AI	Value for AI1 in engineering units (4-byte float - REAL). To access the value, define the variable <i>AT%IDxx</i> . Direct access to %IDxx is not possible.
36, 40	MIX_x_2_AI, MIX_x_3_AI	Value for AI2, AI3
64	MIX_x_y_AO_ACTUAL	Value actually written to AO – clamped to 0-100%.
80	MIX_x_1_STATE, MIX_x_2_STATE	Current status of HSC1 (in bit 0) to HSC2 (in bit 1).
100	MIX_x_INVOLT	Value for Input Voltage in engineering units (4-byte float – REAL – 0 to 32V range).
104	MIX_x_DP	Differential Pressure – REAL – In units of Inches of Water (inH2O) or PSI. The units used depend on the range type of the connected transmitter (see chart below).
108	MIX_x_SP	Static Pressure – REAL – In units of Pounds per Square Inch.
112	MIX_x_RTd	RTD reading – REAL – In units of Degrees Centigrade.
116	MIX_x_TEMP	Estimated Sensor Temperature – REAL – In units of Degrees Centigrade.
120	MIX_x_SENSORID	Sensor ID (Block Number) –INT
122	MIX_x_SENSORSERIES	Sensor Series – SINT
124	MIX_x_SENSORTYPE	Sensor Type – SINT
125	MIX_x_DPRANGE	DP Range Code – SINT

Offset	Default Variable Name where x = board slot number	Description
126	MIX_x_SPRANGE	SP Range Code – SINT
127 (Bit 0)	MIX_x_DPENABLE	Differential Pressure Processing is enabled.
128 (Bit 0)	MIX_x_SPENABLE	Static Pressure Processing is enabled.
129 (Bit 0)	MIX_x_RTDENABLE	RTD Processing is enabled.
132	MIX_x_LASTOPERR	DWORD – Specific Error for last operation – See codes below.
136	MIX_x_ALTERVOLT	This is the alternate input voltage. (For platforms that support alternate voltage input.)
140	MIX_x_MSPTMPT	REAL – Estimated Temperature of the MSP on the CPU board – this is used (by the MSP) to control the battery charging circuit.

Board Type Code:

Board Type	Device ID	Description
1	4 (GFC)	2DI, 2DO, 2HSC, 6 volt
2	4	2DI, 2DO, 2HSC, 3AI, 6 volt
3	4	2DI, 2DO, 2HSC, 3AI, 1AO, 12 volt
4	4	2DI, 2DO, 2HSC, 12 volt
1	7 (3820 XFC)	2DI, 2DO, 2HSC
3	7	2DI, 2DO, 2HSC, 3AI, 1AO
4	7	None
5	7	2DI, 4DO, 2HSC
6	7	2DI, 4DO, 2HSC, 3AI, 1AO
7	9 (Express – CPU)	2HSC, Wet End, RTD, Battery
8	9	2HSC, Battery
9	A (Express – I/O)	2DIO, 4DI, 2DO, 2HSC
A	A	2DIO, 4DI, 2DO, 2HSC, 3AI
B	A	2DIO, 4DI, 2DO, 2HSC, 3AI, 1AO
C	7 (3820 XFC)	2DI, 4DIO, 2HSC
D	7	2DI, 4DIO, 2HSC, 3AI, 1AO

Transmitter types:

Sensor Type	DP Range Code	Range / Units
32 (Differential Pressure)	12	150 InH2O
32 (Differential Pressure)	13	100 InH2O
32 (Differential Pressure)	14	300 InH2O
32 (Differential Pressure)	20	25 psi
12 (Gauge Pressure)	14	300 InH2O
12 (Gauge Pressure)	20	25 psi
12 (Gauge Pressure)	22	100 psi
12 (Gauge Pressure)	23	300 psi
12 (Gauge Pressure)	25	1000 psi
12 (Gauge Pressure)	26	3000 psi
12 (Gauge Pressure)	28	2000 psi
12 (Gauge Pressure)	29	4000 psi

Note: If DP Range Code is < 20, units are InH2O, if >= 20, units are psi.

Sensor Type	SP Range Code	Range / Units
32 (Differential Pressure)	1	1000 psi
32 (Differential Pressure)	2	2000 psi
32 (Differential Pressure)	3	500 psi
32 (Differential Pressure)	4	4000 psi

Specific Error Codes (this is an addition of the following bit values:

0x00000001 Could not erase MSP Info memory
 0x00000002 Could not write MSP Info memory
 0x00000004 Checksum failure in Serial EEP.
 0x00000008 Flash checksum failure in Sensor.
 0x00000010 Flash checksum failure in MSP Info.
 0x00000100 DP - Could not initialize the SA.
 0x00000200 DP - SA flash checksum error.
 0x00000400 DP - Data error after Init done.
 0x00000800 DP - Data error during scan.
 0x00001000 DP - Data scan error.
 0x00010000 SP - Could not initialize the SA.
 0x00020000 SP - SA flash checksum error.
 0x00040000 SP - Data error after Init done.
 0x00080000 SP - Data error during scan.
 0x00100000 SP - Data scan error.
 0x01000000 Could not initialize the ADS.
 0x02000000 Data scan error.
 0x04000000 RTD out of range.
 0x08000000 Temperature calculation error.

Output Map: Size: 152 bytes

Offset	Default Variable Name where x = board slot number y=point number	Description
0	MIX_x_1_DO, MIX_x_2_DO, MIX_x_3_DO, MIX_x_4_DO, MIX_x_5_DO, MIX_x_6_DO	Outputs. 1 bit per value. DO1 is LSB; DO6 is MSB. Typically specified as %QXy.z, where y is I/O space offset, and z is bit number from 0 to 5.

Offset	Default Variable Name where x = board slot number y =point number	Description
2	MIX_x_y_RESET COUNT	HSC 1 = Bit 0, HSC2 = Bit 1 TRUE = Reset counts to 0 after a warm start; FALSE= do NOT reset counts (default).
4	MIX_x_NOINIT	Bit 1. When set TRUE, board is not re-initialized.
6	MIX_x_y_HSC_SEL	1 ms filter select: HSC 1 = Bit 0, HSC2 = Bit 1. Bit set TRUE = 1 ms filter, FALSE=10K Hz counter, default = FALSE.
8	MIX_x_1_AI_ZERO	Zero for AI1 (4-byte float - REAL). To access the value, define the variable AT %QDxx. This variable can be initialized at declaration.
12	MIX_x_1_AI_SPAN	Span for AI1 (4-byte float). If zero, the AI will be scaled from 0 to 100.0.
16, 24	MIX_x_2_AI_ZERO, MIX_x_3_AI_ZERO	Zeros for AI2, AI3
20, 28	MIX_x_2_AI_SPAN, MIX_x_3_AI_SPAN	Spans for AI2, AI3
72	MIX_x_1_AO_ZERO	Zero for AO1 (4-byte float - REAL). To access the value, define the variable AT %QDxx. This variable can be initialized at declaration.
76	MIX_x_1_AO_SPAN	Span for AO1 (4-byte float). If zero, the AO will be scaled from 0 to 100.0.
80	MIX_x_1_AO	Value for AO1 (4-byte float).
120 (Bit 0)*	MIX_x_RESTOREDP	Restore Factory Calibration Values for Differential Pressure.
121 (Bit 0)*	MIX_x_RESTORESP	Restore Factory Calibration Values for Static Pressure.
122 (Bit 0)*	MIX_x_RESTORERTD	Restore Factory Values for RTD.
123 **	MIX_x_CALIBOP	Perform Calibration Operation – SINT 1 DP Zero 2 DP Span 3 SP Zero 4 SP Span 5 RTD Zero (100 Ohms) 6 RTD Span (300 Ohms) 7 RTD Coefficients (A, B, R0) 8 RTD Span (not using 300 Ohms)
128 **	MIX_x_DPSPAN	DP Applied at DP Span Calibration – REAL – Units of Inches of Water..
132 **	MIX_x_SPSPAN	SP Applied at SP Span Calibration – REAL – Units of Pounds per Square Inch.
136 ***	MIX_x_RTDA	RTD Coeff – A
140 ***	MIX_x_RTDB	RTD Coeff – B
144 ***	MIX_x_RTDRO	RTD Coeff – R0
148	MIX_x_RTDSPAN	The applied temperature when calibration operation 8 was performed. (RTD 8)

* Value is cleared by the driver when the operation completes.

** Value is read from the hardware at startup and after any calibration operation. User changes to this value only can be made when Bit 7 of Offset 0 in the input map is set.

*** Same as **; but, reserved for future expansion.

Note:

For any REAL variables, a variable needs to be defined at the proper address; direct access to the memory location via %IDxx is not possible.

Notes for Configuring DI/DO Points on the CWM_RTU board

For This I./O Point:	Configure this Digital Pin in I/O Configurator:	Default Variable Name:	Notes:
DI1	PIN 1:	MIX_1_1_DI	Fixed input – always present
DI2	PIN 2	MIX_1_2_DI	Fixed input – always present
DI3/DO1 is used as a DI	PIN 3	MIX_1_3_DI	When used, do NOT use PIN13.
DI3/DO1 is used as a DO	PIN 13	MIX_1_1_DO	When used, do NOT use PIN3.*
DI4/DO2 is used as a DI	PIN 4	MIX_1_4_DI	When used, do NOT use PIN14.
DI4/DO2 is used as a DO	PIN 14	MIX_1_2_DO	When used, do NOT use PIN4.*
DI5/DO3 is used as a DI	PIN 5	MIX_1_5_DI	When used, do NOT use PIN15.
DI5/DO3 is used as a DO	PIN 15	MIX_1_3_DO	When used, do NOT use PIN5.*
DI6/DO4 is used as a DI	PIN 6	MIX_1_6_DI	When used, do NOT use PIN16.
DI6/DO4 is used as a DO	PIN 16	MIX_1_4_DO	When used, do NOT use PIN6.*

Note: Pins 7 through 12 are unused on this platform.

* Advanced users may optionally read the DI pin, which will be 'TRUE' when the DO is ON; and 'FALSE' when the DI is OFF. Users should never physically wire a point as a DI and then attempt to drive the DO pin, however, as this will override the incoming DI value.

Local I/O – ControlWave Express and ControlWave GFC

CWM_ECPCU High Speed Counter/ RTD/ Wet End Interface Board

This card contains the following I/O: HSC2, Battery Voltage, RTD, and Wet-End interface.

Note:

Due to the design of the SPI interface connecting the CPU to the I/O board, the I/O can only be read / written at one second intervals.

DRIVER_NAME 'CWM_ECPCU'
DATA_TYPE DWORD
DRIVER_PAR1 slot number – not used, specify as 0.
Input Map: Max Size: 144 bytes

Offset	Default Variable Name where x = board slot number	Description
0 (Bit 0)	ECPU_x_BOARDSTATUS	Board status. If this “board” is present, will be FALSE; TRUE indicates that the MSP on the CPU board is not functioning.
1 (Bit 1)	ECPU_x_PSRDERR	Pressure Reading Error. If set, pressure reading is not valid.
1 (Bit 2)	ECPU_x_PSRDINITERR	Pressure Reading – Initialization failure – no retry performed on this error.
1 (Bit 3)	ECPU_x_PSRDCOMMERR	Pressure Reading – Communications failure – will be retried.
1 (Bit 4)	ECPU_x_PSRDNORESET	Pressure Reading – Wet End has been changed without power reset.
1 (Bit 5)	ECPU_x_RTDRDERR	RTD Reading Error.
1 (Bit 6)	ECPU_x_BATRDERR	Battery Voltage Reading Error.
1 (Bit 7)	ECPU_x_ALLOWCALIB	Calibration Commands Allowed. Until this bit is set, all calibration commands are ignored.
2 (Bit 0)	ECPU_x_LASTCALBOP	Set if last calibration or reset operation failed.
3	ECPU_x_BOARDPRESENT	I/O board present. 7 – 2HSC, Wet End, RTD, Battery 8 – 2HSC, Battery.
12	ECPU_x_TIMESTAMP	Timestamp of last sample from HSC. This is the number of Milliseconds since boot.
16	ECPU_x_1_COUNTER	Number of counts since COLD start or clear (Channel 1)
20	ECPU_x_2_COUNTER	Counts for Channel 2
80	ECPU_x_1_STATE	Current status of HSC1 (in bit 0) to HSC2 (in bit 1).
100	ECPU_x_INVOLT	Value for Input Voltage in engineering units (4-byte float – REAL – 0 to 32V range).
104	ECPU_x_DP	Differential Pressure – REAL – In units of Inches of Water (inH2O) or PSI. The units used depend on the range type of the connected transmitter (see chart below).
108	ECPU_x_SP	Static Pressure – REAL – In units of Pounds per Square Inch.

Offset	Default Variable Name where x = board slot number	Description
112	ECPU_x_RTD	RTD reading – REAL – In units of Degrees Centigrade.
116	ECPU_x_TEMP	Estimated Sensor Temperature – REAL – In units of Degrees Centigrade.
120	ECPU_x_SENSORID	Sensor ID (Block Number) –UINT
122	ECPU_x_SENSORSERIES	Sensor Series – SINT
124	ECPU_x_SENSORTYPE	Sensor Type – SINT
125	ECPU_x_DPRANGE	DP Range Code – SINT
126	ECPU_x_SPRANGE	SP Range Code – SINT
127 (Bit 0)	ECPU_x_DPENABLE	Differential Pressure Processing is enabled.
128 (Bit 0)	ECPU_x_SPENABLE	Static Pressure Processing is enabled.
129 (Bit 0)	ECPU_x_RTDENABLE	RTD Processing is enabled.
132	ECPU_x_LASTOPERR	DWORD – Specific Error for last operation – See codes below.
136	ECPU_x_ALTERVOLT	REAL – Voltage at secondary battery input. Not supported, will always be 0.
140	ECPU_x_MSPTEMP	REAL – Estimated Temperature of the MSP on the CPU board – the MSP uses this to control the battery charging circuit.

Transmitter types:

Sensor Type	DP Range Code	Range / Units
32 (Differential Pressure)	12	150 InH20
32 (Differential Pressure)	13	100 InH20
32 (Differential Pressure)	14	300 InH20
32 (Differential Pressure)	20	25 PSI
12 (Gauge Pressure)	14	300 InH20
12 (Gauge Pressure)	20	25 PSI
12 (Gauge Pressure)	22	100 PSI
12 (Gauge Pressure)	23	300 PSI
12 (Gauge Pressure)	25	1000 PSI
12 (Gauge Pressure)	28	2000 PSI

Note: If DP Range Code is < 20, units are InH20, if >= 20, units are PSI.

Specific Error Codes (this is an addition of the following bit values:

0x00000001	Could not erase MSP Info memory
0x00000002	Could not write MSP Info memory
0x00000004	Checksum failure in Serial EEPROM.

0x00000008	Flash checksum failure in Sensor.
0x00000010	Flash checksum failure in MSP Info.
0x00000100	DP - Could not initialize the SA.
0x00000200	DP – SA flash checksum error.
0x00000400	DP - Data error after Init done.
0x00000800	DP - Data error during scan.
0x00001000	DP - Data scan error.
0x00010000	SP - Could not initialize the SA.
0x00020000	SP - SA flash checksum error.
0x00040000	SP - Data error after Init done.
0x00080000	SP - Data error during scan.
0x00100000	SP - Data scan error.
0x01000000	Could not initialize the ADS.
0x02000000	Data scan error.
0x04000000	RTD out of range.
0x08000000	Temperature calculation error.

Output Map: Size: 148 bytes

Offset	Default Variable Name where x = board slot number	Description
2	ECPU_x_1_RESET_COUNT	HSC reset flags: 1 bit per point. HSC1 is LSB (bit 0) , HSC2 is MSB (bit1)
4.1	ECPU_x_NOINIT	Retain HSC values on Warm Start. If variable is set, the HSC counts will be maintained across an application WARM start. If clear, HSC counts will be cleared on both Cold and Warm starts.
120 (Bit 0)*	ECPU_x_RESTOREDP	Restore Factory Calibration Values for Differential Pressure.
121 (Bit 0)*	ECPU_x_RESTORESP	Restore Factory Calibration Values for Static Pressure.
122 (Bit 0)*	ECPU_x_RESTORERTD	Restore Factory Values for RTD.
123 **	ECPU_x_CALIBOP	Perform Calibration Operation – SINT 1 DP Zero 2 DP Span 3 SP Zero 4 SP Span 5 RTD Zero (100 Ohms) 6 RTD Span (300 Ohms)

Offset	Default Variable Name where x = board slot number	Description
		7 RTD Coefficients – Update 8 RTD Span (not using 300 Ohms)
128 **	ECPU_x_DPSPAN	DP Applied at DP Span Calibration – REAL – Units of Inches of Water.
132 **	ECPU_x_SPSPAN	SP Applied at SP Span Calibration – REAL – Units of Pounds per Square Inch.
136 ***	ECPU_x_RTDA	RTD Coeff – A
140 ***	ECPU_x_RTDB	RTD Coeff – B
144 ***	ECPU_x_RTDR0	RTD Coeff – R0
148	ECPU_x_RTDSPAN	The applied temperature when calibration operation 8 was performed. (RTD 8)

* The driver clears value when the operation completes.

** Value is read from the hardware at startup and after any calibration operation. User changes to this value only can be made when Bit 7 of Offset 0 in the input map is set.

*** Same as **; but, reserved for future expansion.

Note:

For any REAL variables, a variable needs to be defined at the proper address; direct access to the memory location via %IDxx is not possible.

CWM_EIO – Mixed I/O Board

This card contains the following I/O: DIO2, DI4, DO2, HSC2, AI3 (Opt), AO1 (Opt).

Note:

The I/O on this board may be read / written at rates up to 20 times per second (50 Millisecond intervals).

DRIVER_NAME 'CWM_EIO'
DATA_TYPE DWORD
DRIVER_PAR1 slot number – not used – specify as 0.
Input Map: Max Size: 120 bytes

Offset	Default Variable Name where x = board slot number	Description
0 (Bit 0)	EIO_x_BOARDSTATUS	Board status. If this I/O board is present, will be FALSE; TRUE indicates that the board is not plugged in.

Offset	Default Variable Name where x = board slot number	Description
0 (Bit 2)	EIO_x_CALIBRATE	Calibration Error. AI and/or AO calibration data is invalid.
0 (Bit 3)	EIO_x_TIMEOUT	Timeout occurred communicating with I/O board.
3	EIO_x_BOARDPRESENT	I/O board present. 9 – 2DIO, 4DI, 2DO, 2HSC 10 – 2DIO, 4DI, 2DO, 2HSC, 3AI. 11 – 2DIO, 4DI, 2DO, 2HSC, 3AI, 1AO
4	EIO_x_1_AI_OUTRANGE	AI under / over range. One bit per point: AI1 in bit 0, AI3 in bit 2.
6	EIO_x_1_AO_OUTRANGE	AO under / over range. AO1 is in bit 0.
8	EIO_x_1_DI EIO_x_2_DI EIO_x_3_DI EIO_x_4_DI	Current status of DI1 (in bit 0) to DI4 (in bit 3). DIO1 as DI (in bit 4), DIO2 as DI (in bit5)
10	EIO_x_1_DO_I EIO_x_2_DO_I	Read-back of current value for DO1 (in bit 0) to DO2 (in bit 1). DIO1 as DO (in bit 2), DIO2 as DO (in bit 3).
12	EIO_x_TIMESTAMP	Timestamp of last sample from HSC. This is the number of milliseconds since boot.
16	EIO_x_1_COUNTER	Number of counts since COLD start or clear (Channel 1)
20	EIO_x_2_COUNTER	Counts for Channel 2
32	EIO_x_1_AI	Value for AI1 in engineering units (4-byte float - REAL). To access the value, define the variable <i>AT %IDxx</i> . Direct access to %IDxx is not possible.
36, 40	EIO_x_2_AI EIO_x_3_AI	Value for AI2, AI3
64	EIO_x_1_AO_ACTUAL	Value actually written to AO1– clamped to 0-100%.
80	EIO_x_1_STATE	Current status of HSC1 (in bit 0) to HSC2 (in bit 1).

Output Map:

Size: 84 bytes

Offset	Default Variable Name where x = board slot number	Description
0	EIO_x_1_DO EIO_x_2_DO	Outputs. 1 bit per value. DO1 is bit 0; DO2 is bit 1, DIO1 as DO is bit2, DIO2 as DO is bit3. Typically specified as %QXy.z, where y is I/O space offset, and z is bit number from 0 to 3.
2	EIO_x_1_RESET_COUNT	HSC reset flags: 1 bit per point. HSC1 is LSB (bit 0) , HSC2 is MSB (bit 1)
4.1	EIO_x_NOINIT	Retain HSC values on Warm Start. If variable is set, the HSC counts will be maintained across an application WARM start. If clear, HSC counts will be cleared on both Cold and Warm starts.
8	EIO_x_1_AI_ZERO	Zero for AI1 (4-byte float - REAL). To access the value, define the variable <i>AT %QDxx</i> . This variable can be initialized at declaration.
12	EIO_x_1_AI_SPAN	Span for AI1 (4-byte float). If zero, the AI will be scaled from 0 to 100.0.
16, 24	EIO_x_2_AI_ZERO EIO_x_3_AI_ZERO	Zeros for AI2, AI3
20, 28	EIO_x_2_AI_SPAN EIO_x_3_AI_SPAN	Spans for AI2, AI3

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
32, 40		F, use 32.0
12, 20, 28, 36, 44	TC6_x_y_SPAN	Spans for TC2, TC3, TC4, to TC6 – Example for C to F, use 1.8
64	TC6_x_y_MODE	Point type for TC1; see Thermocouple type codes section for details.
65, 66, 67, 68, 69	TC6_x_y_MODE	Point types for TC2, TC3, TC4, TC5, and TC6.

Type codes for Thermocouple Points.

Type Code	Code	Range
0	B	Thermocouple: 100C – +1820C
1	E	Thermocouple: -270C – +1000C
2	J	Thermocouple: -210C – +1200C
3	K	Thermocouple: -270C – +1370C
4	R	Thermocouple: -50C – +1720C
5	S	Thermocouple: -50C – +1760C
6	T	Thermocouple: -270C – +400C
7	Unused	Unused
8	10MV	Voltage Inputs: -10 mV to +10 mV (Outputs as 0.0 to 1.0)
9	C	Thermocouple: 0C – +2315C
10	N	Thermocouple: -270C – +1300C

CWM_RTD4 – ControlWave Micro – 4 Point Resistance Temperature Device (RTD) Board

DRIVER_NAME 'CWM_RTD4'

DATA_TYPE DWORD (32 bits)

DRIVER_PAR1 slot number.

Input Map: Max Size: 24 bytes

Due to the amount of time required to process the RTD points, it is highly recommended that this I/O driver be assigned to a task (instead of “No Task”). Care should also be taken in using the I/O board in a task of less than 40ms.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0 (Bit 0)	RTD4_x_BOARDSTATUS	If set, the board is not present.
0 (Bit 2)	RTD4_x_CALIBRATE	If set indicates invalid calibration data written to the board.
0 (Bit 3)	RTD4_x_TIMEOUT	If set, indicates that had an error reading or writing

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
		to the board.
1 (Bit 0)	RTD4_x_LASTCALBOP	Set if last calibration or reset operation failed.
1 (Bit 7)	RTD4_x_CALBCMD	Calibration Commands Allowed. Until this bit is set, all calibration commands are ignored.
4	RTD4_x_y_READERR	RTD Reading Error. Bit 0 is RTD1, Bit 3 is RTD4
8	RTD4_x_y	RTD1 reading – REAL – In units of Degrees Centigrade (unless scaled by values in the output map).
12, 16, ...20	RTD4_x_y	Readings for RTD2 ... RTD4.

Output Map:

Max Size: 200 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RTD4_x_y_ZERO	Zero for RTD 1(4-byte float - REAL). Example: for C to F, use 32.0. Defaults to 0.0
4	RTD4_x_y_SPAN	Span for RTD 1(4-byte float – REAL). If zero, RTD will not be scaled. If specified, the scaled value will be $ORG_VALUE * Span + Zero$. Example for C to F, use 1.8.
8,16,24,32 40,48,56	RTD4_x_y_ZERO	Zero for RTDs 2-8
12,20,28, 36,44,52, 60	RTD4_x_y_SPAN	Span for RTDs 2-8
64	RTD4_x_y_MODE	NOT USED – FOR EXPANSION
65,66,67, 68,69	RTD4_x_y_MODE	NOT USED – FOR FUTURE EXPANSION
100		NOT USED – FOR FUTURE EXPANSION
120 (Bit 0) *	RTD4_x_y_RESTORE	If set, restore RTD 1 calibration to Factory Defaults. Will be reset when operation completes.
121 *	RTD4_x_y_OPERATION	Calibration Operation for RTD 1 – SINT 5 RTD Zero (100 Ohms) 6 RTD Span (300 Ohms) 7 RTD Coefficients (A, B, R0) 8 RTD Span (not using 300 Ohms)
124**	RTD4_x_y_COEFF_A	Coefficient A (RTD 1)
128**	RTD4_x_y_COEFF_B	Coefficient B (RTD 1)
132**	RTD4_x_y_COEFF_R0	Coefficient R0 (RTD 1)
136**	RTD4_x_y_APPLIED	The applied temperature when calibration operation 8 was performed. (RTD 1)
140 (Bit 0) *	RTD4_x_y_RESTORE	If set, restore RTD 2 calibration to Factory Defaults. Will be reset when operation completes.
141 *	RTD4_x_y_OPERATION	Calibration Operation for RTD 2 – SINT

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
		5 RTD Zero (100 Ohms) 6 RTD Span (300 Ohms) 7 RTD Coefficients (A, B, R0) 8 RTD Span (not using 300 Ohms)
144 **	RTD4_x_y_COEFF_A???	Coefficient A (RTD 2)
148 **	RTD4_x_y_COEFF_B???	Coefficient B (RTD 2)
152 **	RTD4_x_y_COEFF_R0???	Coefficient R0 (RTD 2)
156 **	RTD4_x_y_APPLIED????	The applied temperature when calibration operation 8 was performed. (RTD 2)
....	
180 (Bit 0) *	RTD4_x_y_RESTORE	If set, restore RTD 4 calibration to Factory Defaults. Will be reset when operation completes.
181 *	RTD4_x_y_OPERATION	Calibration Operation for RTD 4 – SINT 5 RTD Zero (100 Ohms) 6 RTD Span (300 Ohms) 7 RTD Coefficients (A, B, R0) 8 RTD Span (not using 300 Ohms)
184 **	RTD4_x_y_COEFF_A???	Coefficient A (RTD 4)
188 **	RTD4_x_y_COEFF_B???	Coefficient B (RTD 4)
192 **	RTD4_x_y_COEFF_R0???	Coefficient R0 (RTD 4)
196 **	RTD4_x_y_APPLIED????	The applied temperature when calibration operation 8 was performed. (RTD 4)

* Value written to perform operation. The value will be reset by driver when the operation completes.

** Value is read from the board by the driver. In order to perform calibration operations 7 and 8, the user can overwrite the values; then, issue the calibration command.

Local I/O – ControlWave CW10, CW30, CW35

Note:

I/O boards of type CXX (CW10 and CW30) cannot be mixed within the same ControlWave Designer resource with I/O boards for any other ControlWave Platform. This can, however, be done with two resources within the same ControlWave Designer project.

CXX_AI8 – ControlWave CW_10/CW_30/CW_35 4- or 8-Analog Input Pin Board

DRIVER_NAME 'CXX_AI8'
DATA_TYPE DWORD (32 bits)
DRIVER_PAR1 slot number.

Input Map: Max Size: 40 bytes

Due to the amount of time required to process the AI points, it is highly recommended that this I/O driver be assigned to a task (instead of "No Task"). Care should also be taken in using the I/O board in a task of less than 25ms.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0 (Bit 0)	AI8_x_BOARDSTATUS	Board status. Bit 0 is set if board is not present.
0 (Bit 1)	AI8_x_LASTOPERATION	Board status. Bit 1 is set if the last conversion operation failed.
4	AI8_x_y_OUTRANGE	1 bit per AI, AI1 is bit 0, AI8 is bit 7. If set, input is Out-of-range.
8	AI8_x_y	Value for AI1 in engineering units (4-byte float - REAL). To access the value, define the variable AT %IDxx. Direct access to %IDxx is not possible.
12, 16, ...	AI8_x_y	Value for AI2, AI3, ...

Output Map: Max Size: 64 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	AI8_x_y_ZERO	Zero for AI1 (4-byte float - REAL). To access the value, define the variable AT %QDxx. This variable can be initialized at declaration.
4	AI8_x_y_SPAN	Span for AI1 (4-byte float). If zero, the AI will be scaled from 0 to 100.0.
8, 16, 24, ...	AI8_x_y_ZERO	Zeros for AI2, AI3, ...
12, 20, 28, ...	AI8_x_y_SPAN	Spans for AI2, AI3, ...

CXX_AO4 – ControlWave CW_10/CW_30/CW_35 2 or 4 Analog Output Pin Board

DRIVER_NAME 'CXX_AO4'
 DATA_TYPE DWORD (32 bits)
 DRIVER_PAR1 slot number.

Input Map: Max Size: 24 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0 (bit 0)	AO4_x_BOARDSTATUS	Board status. Bit 0 is set if board is not present.
0 (bit 1)	AO4_x_LASTOPERATION	Bit 1 is set if the last conversion operation failed.
4	AO4_x_y_OUTRANGE	1 bit per AO, AO1 is bit 0, AO4 is bit 3. If set, output is Out-of-range.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
8,12,16,20	AO4_x_y_ACTUAL	Real value of value actually output. Clamped to 0-100% of scale.

Output Map: Max Size: 48 bytes

Due to the amount of time required to process the AO points, it is highly recommended that this I/O driver be assigned to a task (instead of "No Task"). Care should also be taken in using the I/O board in a task of less than 25ms.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	AO4_x_y_ZERO	Zero for AO1 (4-byte float - REAL). To access the value, define the variable <i>AT%QDxx</i> . This variable can be initialized at declaration.
4	AO4_x_y_SPAN	Span for AO1 (4-byte float). If zero, the AO will be scaled from 0 to 100.0.
8	AO4_x_y	Value for AO1 (4-byte float).
12, 24, 36	AO4_x_y_ZERO	Zeros for AO2, AO3, AO4
16, 28, 40	AO4_x_y_SPAN	Spans for AO2, AO3, AO4
20, 32, 44	AO4_x_y	Values for AO2, AO3, AO4

CXX_DI16 – ControlWave CW_10/CW_30/CW_35 4 or 8 or 16 Digital Input Pin Board

DRIVER_NAME 'CXX_DI16'

DATA_TYPE DWORD (32 bits)

DRIVER_PAR1 slot number.

DRIVER_PAR2 Mask of points enabled as low-speed counters

Input Map: Max Size: 84 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	DI16_x_BOARDSTATUS	Board status.
4	DI16_x_y	Current status of DI1 (in bit 0) to DI8 (in bit 7).
5	DI16_x_y	Current status of DI9 (in bit 0) to DI16 (in bit 7).
16	DI16_x_y_TIMESTAMP	Timestamp of last sample from HSC. This is the number of milliseconds since boot.
20	DI16_x_y_COUNTER	32-bit counter for DI1
24, ...	DI16_x_y_COUNTER	32-bit counters for DI2 – DI16

Output Map: Size: 4 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	DI16_x_y_RESET_COUNT	Counter reset flags, DI1 (in bit 0) to DI8 (in bit 7)
1	DI16_x_y_RESET_COUNT	Counter reset flags, 9 to 16.
2.1	DI16_x_y_NOINIT	Maintain Counts across Warm Start (TRUE = maintain)

CXX_DO16 – ControlWave CW_10/CW_30/CW_35 4- or 8- or 16-Digital Output Pin Board

DRIVER_NAME 'CXX_DO16'

DATA_TYPE BYTE

DRIVER_PAR1 slot number.

Input Map: Max Size: 6 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	DO16_x_BOARDSTATUS	Board status.
4	DO16_x_y_l	Read-back of output: 1 bit per value. DO1 is LSB, DO8 is MSB.
5	DO16_x_y_l	Read-back for outputs 9 to 16.

Output Map: Size: 2 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	DO16_x_y	Outputs. 1 bit per value. DO1 is LSB; DO8 is MSB. Typically specified as %QXy.z, where y is I/O space offset, and z is bit number from 0 to 7.
1	DO16_x_y	Outputs 9 to 16.

CXX_HSC8 – ControlWave CW_10/CW_30/CW_35 4 or 8 Channel High Speed Counter Board

DRIVER_NAME 'CXX_HSC8'

DATA_TYPE DWORD (32 bits)

DRIVER_PAR1 slot number.

Input Map: Max Size: 40 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	HSC8_x_BOARDSTATUS	Board status. Only bit 0 is currently defined. If set, board is not present.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
4	HSC8_x_TIMESTAMP	Timestamp of last sample from HSC. This is the number of milliseconds since boot.
8	HSC8_x_y_COUNTER	Number of counts since boot (Channel 1)
12, 16, ..	HSC8_x_y_COUNTER	Counts for Channel 2, 3, etc.

Output Map:

Size: 4 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	HSC8_x_y_RESET_COUNT	Counter reset flags, HSC1 (in bit 0) to HSC8 (in bit 7)
2.1	HSC8_x_NOINIT	Maintain Counts across Warm Start (TRUE = maintain)

CXX_LL4 – ControlWave CW_10/CW_30/CW_35 4 Low-Level Analog Input Pin Board

DRIVER_NAME 'CXX_LL4'

DATA_TYPE DWORD (32 bits)

DRIVER_PAR1 slot number.

Input Map: Max Size: 24 bytes

Due to the amount of time required to process the Low-level points, it is highly recommended that this I/O driver be assigned to a task (instead of “No Task”). Care should also be taken in using the I/O board in a task of less than 25ms.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0 (bit 0)	LL4_x_BOARDSTATUS	Board status. Bit 0 is set if board is not present.
0 (bit 1)	LL4_x_LASTOPERATION	Board status. Bit 1 is set if the board is either in calibration mode or being reset.
4	LL4_x_y_OUTRANGE	1 bit per LL, LL1 is bit 0, LL4 is bit 3. If set, input is Out-of-range.
6		BYTE value – Board error code – see table below. * Note: a variable is not automatically created by the System Variable Wizard for this field.
8	LL4_x_y	Value for LL1 in engineering units (4-byte float - REAL). To access the value, define the variable AT %IDxx. Direct access to %IDxx is not possible.
12, 16, 20	LL4_x_y	Value for LL2, LL3, LL4

LLAI Board Error Codes (values are in Hex 16#):

0n Channel errors – Bit 0 = point 1, Bit 1 = point 2, etc.

10 Board is in process of being reset.

An Board is in Calibration mode.

Output Map: Max Size: 68 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	LL4_x_y_ZERO	Zero for LL1 (4-byte float - REAL). To access the value, define the variable AT %QDxx. This variable can be initialized at declaration.
4	LL4_x_y_SPAN	Span for LL1 (4-byte float). If zero, the LL will be scaled as in the chart below. If specified, the new value will be $ORG_VALUE * Span + Zero$.
8, 16, 24	LL4_x_y_ZERO	Zeros for LL2, LL3, LL4 – Example: for C to F, use 32.0
12, 20, 28	LL4_x_y_SPAN	Spans for LL2, LL3, LL4 – Example for C to F, use 1.8
64	LL4_x_y_MODE	Point type for LL1, see table below for type codes.
65, 66, 67	LL4_x_y_MODE	Point types for LL2, LL3, and LL4.

Type codes for Low-Level Points

Type Code	Code	Range
0	B	Thermocouple: 100C – +1820C
1	E	Thermocouple: -270C – +1000C
2	J	Thermocouple: -210C – +1200C
3	K	Thermocouple: -270C – +1370C
4	R	Thermocouple: -50C – +1720C
5	S	Thermocouple: -50C – +1760C
6	T	Thermocouple: -270C – +400C
7	RTD	RTD: -220C to +850C
8	10MV	Voltage Inputs: -10 mV to +10 mV (Outputs as 0.0 to 1.0)

I/O – ControlWave CW_31

Common Status Information

The first two bytes of the input map contain status information, which is common to all Expansion Rack boards.

Byte	Bit	Description
0	0 (0x1, 1)	No Board is present in the destination Rack.
0	3 (0x8, 8)	Board type does not match the board installed in the Rack.
0	4 (0x10, 16)	Communications lost with the Expansion Rack.
0	7 (0x80, 128)	Initial opening of channel to the Expansion Rack has not been completed.

RXX_AI8 – CW_31 4 or 8 Pin Analog Input Board

DRIVER_NAME	'RXX_AI8'
DATA_TYPE	DWORD (32 bits)
DRIVER_PAR1	slot number.
DRIVER_PAR2	First two bytes of Primary IP address
DRIVER_PAR3	Lower two bytes of Primary IP address
Input Map:	Max Size: 40 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RXAI_x_DRIVERSTATUS	Board Status. See Common Status Information section.
2 (bit 1)	RXAI_x_LASTOPERATION	Bit 1 – Last conversion operation failed.
4	RXAI_x_y_OUTRANGE	1 bit per AI, AI1 is bit 0, AI8 is bit 7. If set, input is Out-of-range.
8	RXAI_x_y	Value for AI1 in engineering units (4-byte float - REAL). To access the value, define the variable AT %IDxx. Direct access to %IDxx is not possible.
12, 16, ...	RXAI_x_y	Value for AI2, AI3, ...

Output Map: Max Size: 64 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RXAI_x_y_ZERO	Zero for AI1 (4-byte float - REAL). To access the value, define the variable AT %QDxx. This variable can be initialized at declaration.
4	RXAI_x_y_SPAN	Span for AI1 (4-byte float). If zero, the AI will be scaled from 0 to 100.0.
8, 16, 24, ...	RXAI_x_y_ZERO	Zeros for AI2, AI3, ...
12, 20, 28, ...	RXAI_x_y_SPAN	Spans for AI2, AI3, ...

RXX_AO4 – CW_31 2 or 4 Analog Output Pin Board

DRIVER_NAME	'RXX_AO4'
DATA_TYPE	DWORD (32 bits)
DRIVER_PAR1	slot number.

DRIVER_PAR2 First two bytes of Primary IP address
 DRIVER_PAR3 Lower two bytes of Primary IP address
 Input Map: Max Size: 24 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RXAO_x_DRIVERSTATUS	Board Status. See Common Status Information section.
2 (bit 1)	RXAO_x_LASTOPERATION	Bit 1 – Last conversion operation failed.
4	RXAO_x_y_OUTRANGE	1 bit per AO, AO0 is bit 0, AO4 is bit 3. If set, output is Out-of-range.
8,12,16,20	RXAO_x_y_ACTUAL	Real value of value actually output. Clamped to 0-100% of scale.

Output Map: Max Size: 48 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RXAO_x_y_ZERO	Zero for AO1 (4-byte float - REAL). To access the value, define the variable AT%QDxx. This variable can be initialized at declaration.
4	RXAO_x_y_SPAN	Span for AO1 (4-byte float). If zero, the AO will be scaled from 0 to 100.0.
8	RXAO_x_y	Value for AO1 (4-byte float).
12, 24, 36	RXAO_x_y_ZERO	Zeros for AO2, AO3, AO4
16, 28, 40	RXAO_x_y_SPAN	Spans for AO2, AO3, AO4
20, 32, 44	RXAO_x_y	Values for AO2, AO3, AO4

RXX_DI16 CW_31 8 or 16 Digital Input Pin Board

DRIVER_NAME 'RXX_DI16'
 DATA_TYPE DWORD (32 bits)
 DRIVER_PAR1 slot number.
 DRIVER_PAR2 First two bytes of Primary IP address
 DRIVER_PAR3 Lower two bytes of Primary IP address

Note:

Low speed counter processing is automatically enabled for all points.

Input Map: Max Size: 84 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RXDI_x_DRIVERSTATUS	Board Status. See Common Status Information section.
4	RXDI_x_y	Current status of DI1 (in bit 0) to DI8 (in bit 7).
5	RXDI_x_y	Current status of DI9 (in bit 0) to DI16 (in bit 7).
16	RXDI_x_TIMESTAMP	Timestamp of last sample. This is the number of milliseconds since boot.
20	RXDI_x_y_COUNTER	32-bit counter for DI1
24, ...	RXDI_x_y_COUNTER	32-bit counters for DI2 – DI16

Output Map: Size: 4 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RXDI_x_y_RESET_COUNT	Counter reset flags, DI1 (in bit 0) to DI8 (in bit 7)
1	RXDI_x_y_RESET_COUNT	Counter reset flags, 9 to 16.
2 (bit 1)	RXDI_x_NOINIT	Maintain Counts across Warm Start (TRUE = maintain)

RXX_DO16 CW_31 4, 8 or 16 Digital Output Pin Board

DRIVER_NAME 'RXX_DO16'
 DATA_TYPE BYTE
 DRIVER_PAR1 slot number.
 DRIVER_PAR2 First two bytes of Primary IP address
 DRIVER_PAR3 Lower two bytes of Primary IP address
 Input Map: Max Size: 6 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RXDO_x_DRIVERSTATUS	Board Status. See Common Status Information section.
4	RXDO_x_y_I	Read-back of output: 1 bit per value. DO1 is LSB, DO8 is MSB.
5	RXDO_x_y_I	Read-back for outputs 9 to 16.

Output Map: Size: 2 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RXDO_x_y	Outputs. 1 bit per value. DO1 is LSB; DO8 is MSB. Typically specified as %QXy.z, where y is I/O space

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
		offset, and z is bit number from 0 to 7.
1	RXDO_x_y	Outputs 9 to 16.

RXX_HSC8 CW_31 4 or 8 Channel High Speed Counter Board

DRIVER_NAME 'RXX_HSC8'
 DATA_TYPE DWORD (32 bits)
 DRIVER_PAR1 slot number.
 DRIVER_PAR2 First two bytes of Primary IP address
 DRIVER_PAR3 Lower two bytes of Primary IP address
 Input Map: Max Size: 40 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RXHSC_x_DRIVERSTATUS	Board Status. See Common Status Information section.
4	RXHSC_x_TIMESTAMP	Timestamp of last sample from HSC. This is the number of milliseconds since boot.
8	RXHSC_x_y_COUNTER	Number of counts since boot (Channel 1)
12, 16, ..	RXHSC_x_y_COUNTER	Counts for Channel 2, 3, etc.

Output Map: Size: 4 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RXHSC_x_y_RESET_COUNT	Counter reset flags, HSC1 (in bit 0) to HSC8 (in bit 7)
2 (bit 1)	RXHSC_x_NOINIT	Maintain Counts across Warm Start (TRUE = maintain)

RXX_LL4 CW_31 4 Low Level Analog Input Pin Board

DRIVER_NAME 'RXX_LL4'
 DATA_TYPE DWORD (32 bits)
 DRIVER_PAR1 slot number.
 DRIVER_PAR2 First two bytes of Primary IP address
 DRIVER_PAR3 Lower two bytes of Primary IP address
 Input Map: Max Size: 24 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RXLL_x_DRIVERSTATUS	Board Status. See Common Status Information section.
2	RXLL_x_LASTOPERATION	Bit 1 - Calibration Mode or hardware is being reset
4	RXLL_x_y_OUTRANGE	1 bit per LL, LL1 is bit 0, LL4 is bit 3. If set, input is Out-of-range.
6		BYTE value – Board error code – see table below. * Note: a variable is not automatically created by the IO Configuration Wizard for this field.
8	RXLL_x_y	Value for LL1 in engineering units (4-byte float - REAL). To access the value, define the variable AT %IDxx. Direct access to %IDxx is not possible.
12, 16, 20	RXLL_x_y	Value for LL2, LL3, LL4

LLAI Board Error Codes (values are in Hex 16#):

0n Channel errors – Bit 0 = point 1, Bit 1 = point 2, etc.

10 Board is in process of being reset.

An Board is in Calibration mode.

Output Map: Max Size: 68 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RXLL_x_y_ZERO	Zero for LL1 (4-byte float - REAL). To access the value, define the variable AT %QDxx. This variable can be initialized at declaration.
4	RXLL_x_y_SPAN	Span for LL1 (4-byte float). If zero, the LL will be scaled as in the chart below. If specified, the new value will be $ORG_VALUE * Span + Zero$.
8, 16, 24	RXLL_x_y_ZERO	Zeros for LL2, LL3, LL4 – Example: for C to F, use 32.0
12, 20, 28	RXLL_x_y_SPAN	Spans for LL2, LL3, LL4 – Example for C to F, use 1.8
64	RXLL_x_y_MODE	Point type for LL1, see table below for type codes.
65, 66, 67	RXLL_x_y_MODE	Point types for LL2, LL3, and LL4.

Type codes for Low-Level Points

Type Code	ACCOL Code	Range
0	B	Thermocouple: 100C – +1820C
1	E	Thermocouple: -270C – +1000C

Type Code	ACCOL Code	Range
2	J	Thermocouple: -210C – +1200C
3	K	Thermocouple: -270C – +1370C
4	R	Thermocouple: -50C – +1720C
5	S	Thermocouple: -50C – +1760C
6	T	Thermocouple: -270C – +400C
7	RTD	RTD: -220C to +850C
8	10MV	Voltage Inputs: -10 mV to +10 mV (Outputs as 0.0 to 1.0)

RXX_STAT CW_31 Status Board

In addition, an Expansion Rack Status board (RXX_STAT) is defined. This board has a similar I/O map as the status board for the ControlWave Expansion Rack (ER_STAT).

DRIVER_NAME	'RXX_STAT'
DATA_TYPE	DWORD (32 bits)
DRIVER_PAR1	slot number (ignored – specify as zero).
DRIVER_PAR2	First two bytes of Primary IP address
DRIVER_PAR3	Lower two bytes of Primary IP address
Input Map:	Max Size: 1068 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	RXSTAT_x_BOARDSTATUS	Board Status. See Common Status Information section.
4 (bit 0)	RXSTAT_x_BATSTAT	Memory battery status at Expansion Rack. TRUE indicates good battery.
4 (bit 1)	RXSTAT_x_HOTCARDSTAT	TRUE indicates that a HOT Card replacement is in progress at the Expansion Rack. Will always be FALSE on platforms which do not support on-line HOT Card replacement.
8	RXSTAT_x_HOTCARDCT	32-Bit Count of Hot Card replacement events which have occurred.
12	RXSTAT_x_DOWNTIMEUSER	32-Bit Count – Number of seconds (as configured by the user) that the Expansion Rack can be powered off before the outputs are reset to defaults when the unit is powered back up.
16	RXSTAT_x_DOWNTIMEACT	32-Bit Count – Number of seconds that the Expansion Rack was powered off on the last power fail.
20	RXSTAT_x_WRITECT	32-Bit Count – Number of I/O updates sent to the Expansion Rack
24	RXSTAT_x_READCT	32-Bit Count – Number of I/O updates sent from the

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
		Expansion Rack
28	RXSTAT_x_CONNECTS	32-Bit Count – Number of Connects and Disconnects made to the Expansion Rack.
32	RXSTAT_x_HEARTBEAT	32-Bit Count – Number of IDLE time Heartbeats sent from the Expansion Rack to the host.
42	RXSTAT_x_BDSTR1	String – I/O board string for slot #1
127	RXSTAT_x_BDSTR2	String – I/O board string for slot #2
212	RXSTAT_x_BDSTR3	String – I/O board string for slot #3
297	RXSTAT_x_BDSTR4	String – I/O board string for slot #4
382	RXSTAT_x_BDSTR5	String – I/O board string for slot #5
467	RXSTAT_x_BDSTR6	String – I/O board string for slot #6
552	RXSTAT_x_BDSTR7	String – I/O board string for slot #7
637	RXSTAT_x_BDSTR8	String – I/O board string for slot #8
724		DINT – Current redundancy status of the Expansion Rack – will always be zero on this platform
728	RXSTAT_x_BDSTR9	String – I/O board string for slot #9
813	RXSTAT_x_BDSTR10	String – I/O board string for slot #10
898	RXSTAT_x_BDSTR11	String – I/O board string for slot #11
983	RXSTAT_x_BDSTR12	String – I/O board string for slot #12
1068	RXSTAT_x_BDSTR13	String – I/O board string for slot #13
1153	RXSTAT_x_BDSTR14	String – I/O board string for slot #14 (Note slot #13 and #14 will be populated with zeroes (NULL string))
1240	RXSTAT_x_INPUTVOLTS	Float – Reading (in volts) of power-supply input voltage.

Output Map: Max Size: 4 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0.0		If this is a Redundant Expansion Rack, and the Standby is valid, then writing a TRUE to this location will cause the rack to fail-over. Not used on this platform.
1.0	RXSTAT_x_RDN_IOERR_WARN	

ControlWave MICRO I/O Expansion Rack

Common Status Information

The first two bytes of the input map contain status information, which is common to all Expansion Rack boards.

Byte	Bit	Description
0	0 (0x1, 1)	No board is present in the destination rack.
0	3 (0x8, 8)	Board type does not match the board installed in the rack.
0	4 (0x10, 16)	Communications lost with the expansion rack.
0	7 (0x80, 128)	Initial opening of channel to the expansion rack has not been completed.

ERM_DO16 – ControlWave MICRO I/O Expansion Rack 16 Digital Output Pin Board

DRIVER_NAME	'ERM_DO16'
DATA_TYPE	BYTE
DRIVER_PAR1	slot number.
DRIVER_PAR2	First two bytes of Primary IP address
DRIVER_PAR3	Lower two bytes of Primary IP address
Input Map:	Max Size: 6 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERDO_x_DRIVERSTATUS	Board Status. See Common Status Information section.
4,5	ERDO_x_y_I	DO status as seen by card. bit per value.

Output Map: Size: 5 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERDO_x_y	Outputs. 1 bit per value. DO1 is LSB; DO8 is MSB. Typically specified as %QXy.z, where y is I/O space offset, and z is bit number from 0 to 7.
1	ERDO_x_y	Outputs 9 to 16.
4	ERDO_x_LEDSTATUS	Single bit. If bit is set, the diagnostic LEDs for the points are turned off to save power.

ERM_DI16 – ControlWave MICRO I/O Expansion Rack 16 Digital Input Pin Board

DRIVER_NAME	'ERM_DI16'
DATA_TYPE	DWORD (32 bits)

DRIVER_PAR1	slot number.
DRIVER_PAR2	First two bytes of Primary IP address
DRIVER_PAR3	Lower two bytes of Primary IP address
Input Map:	Max Size: 8 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERDI_x_DRIVERSTATUS	Board Status. See Common Status Information section.
4	ERDI_x_y	Current status of DI1 (in bit 0) to DI8 (in bit 7).
5	ERDI_x_y	Current status of DI9 (in bit 0) to DI16 (in bit 7).

Output Map: Size: 5 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
4	ERDI_x_LEDSTATUS	Single bit. If bit is set, the diagnostic LEDs for the points are turned off to save power.

ERM_MD ControlWave MICRO I/O Expansion Rack Mixed Digital (12 DI / 4 DO) Board

DRIVER_NAME	'ERM_MD'
DATA_TYPE	BYTE
DRIVER_PAR1	slot number.
DRIVER_PAR2	First two bytes of Primary IP address
DRIVER_PAR3	Lower two bytes of Primary IP address
Input Map:	Max Size: 9 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERDIDO_x_DRIVERSTATUS	Board Status. See Common Status Information section.
4	ERDIDO_x_y	Current status of DI1 (in bit 0) to DI8 (in bit 7).
5	ERDIDO_x_y	Current status of DI9 (in bit 0) to DI12 (in bit 3).
8	ERDIDO_x_y_O_I	DO status as seen by card. 1 bit per value.

Output Map: Size: 5 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERDIDO_x_y_O	Outputs. 1 bit per value. DO1 is LSB; DO4 is MSB. Typically specified as %QXy.z, where y is I/O space offset, and z is bit number from 0 to 3.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
4	ERDIDO_x_LEDSTATUS	Single bit. If bit is set, the diagnostic LEDs for the points are turned off to save power.

ERM_AI8 – ControlWave MICRO I/O Expansion Rack – 8 Analog Input Pin Board

DRIVER_NAME 'ERM_AI8'

DATA_TYPE DWORD (32 bits)

DRIVER_PAR1 slot number.

DRIVER_PAR2 First two bytes of Primary IP address

DRIVER_PAR3 Lower two bytes of Primary IP address

Input Map: Max Size: 40 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERAI_x_DRIVERSTATUS	Board Status. See Common Status Information section.
2 (bit 2)	ERAI_x_CALIBRATE	Bit 2 is set if the calibration data is invalid.
2 (bit 3)	ERAI_x_TIMEOUT	Bit 3 is set to indicate that a board data read has timed out.
4	ERAI_x_y_OUTRANGE	1 bit per AI, AI1 is bit 0, AI8 is bit 7. If set, input is Out-of-range.
8	ERAI_x_y	Value for AI1 in engineering units (4-byte float - REAL). To access the value, define the variable AT %IDxx. Direct access to %IDxx is not possible.
12, 16, ...	ERAI_x_y	Value for AI2, AI3, ...

Output Map: Max Size: 132 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERAI_x_y_ZERO	Zero for AI1 (4-byte float - REAL). To access the value, define the variable AT %QDxx. This variable can be initialized at declaration.
4	ERAI_x_y_SPAN	Span for AI1 (4-byte float). If zero, the AI will be scaled from 0 to 100.0.
8, 16, 24, ...	ERAI_x_y_ZERO	Zeros for AI2, AI3, ...
12, 20, 28, ...	ERAI_x_y_SPAN	Spans for AI2, AI3, ...
64, 72, 80, ...	ERAI_x_y_BOTTOMRANGE	Bottom end of usable current / voltage range for this input. Specified as a REAL. For example, for a 1-5V input, this value is 1.0.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
68, 76, 84, ...	ERAI_x_y_TOPRANGE	Top end of usable current / voltage range for this input. Specified as a REAL. For example, for a 1-5V input, this value is 5.0.
128	ERAI_x_y_MODE	1 bit per AI; set TRUE if the point is voltage; FALSE if current.

ERM_AO4 – ControlWave MICRO I/O Expansion Rack - 4 Analog Output Pin Board

DRIVER_NAME 'ERM_AO4'

DATA_TYPE DWORD (32 bits)

DRIVER_PAR1 slot number.

DRIVER_PAR2 First two bytes of Primary IP address

DRIVER_PAR3 Lower two bytes of Primary IP address

Input Map: Max Size: 24 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERAO_x_DRIVERSTATUS	Board Status. See Common Status Information section.
2 (bit 2)	ERAO_x_CALIBRATE	Bit 2 is set if the calibration data is invalid
2 (bit 3)	ERAO_x_TIMEOUT	Bit 3 is set to indicate that a board data read has timed out.
6	ERAO_x_y_OUTRANGE	1 bit per AO, AO1 is bit 0, AO4 is bit 3. If set, output is Out-of-range.
8,12,16,20	ERAO_x_y_ACTUAL	Real value of value actually output. Clamped to 0-100% of scale.

Output Map: Max Size: 48 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERAO_x_y_ZERO	Zero for AO1 (4-byte float - REAL). To access the value, define the variable <i>AT%QDxx</i> . This variable can be initialized at declaration.
4	ERAO_x_y_SPAN	Span for AO1 (4-byte float). If zero, the AO will be scaled from 0 to 100.0.
8	ERAO_x_y	Value for AO1 (4-byte float).
12, 24, 36	ERAO_x_y_ZERO	Zeros for AO2, AO3, AO4
16, 28, 40	ERAO_x_y_SPAN	Spans for AO2, AO3, AO4
20, 32, 44	ERAO_x_y	Values for AO2, AO3, AO4

ERM_AI6 – ControlWave MICRO I/O Expansion Rack 6 Analog Input Pin Board

DRIVER_NAME	'ERM_AI6'
DATA_TYPE	DWORD (32 bits)
DRIVER_PAR1	slot number.
DRIVER_PAR2	First two bytes of Primary IP address
DRIVER_PAR3	Lower two bytes of Primary IP address
Input Map:	Max Size: 32 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERAI_x_DRIVERSTATUS	Board Status. See Common Status Information section.
2 (bit 2)	ERAI_x_CALIBRATE	Bit 2 is set if the calibration data is invalid.
2 (bit 3)	ERAI_x_TIMEOUT	Bit 3 is set to indicate that a board data read has timed out.
4	ERAI_x_y_OUTRANGE	1 bit per AI, AI1 is bit 0, AI6 is bit 5. If set, input is Out-of-range.
8	ERAI_x_y	Value for AI1 in engineering units (4-byte float - REAL). To access the value, define the variable AT %IDxx. Direct access to %IDxx is not possible.
12, 16, ...	ERAI_x_y	Value for AI2, AI3, ...

Output Map: Max Size: 48 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERAI_x_y_ZERO	Zero for AI1 (4-byte float - REAL). To access the value, define the variable AT %QDxx. This variable can be initialized at declaration.
4	ERAI_x_y_SPAN	Span for AI1 (4-byte float). If zero, the AI will be scaled from 0 to 100.0.
8, 16, 24, ...	ERAI_x_y_ZERO	Zeros for AI2, AI3, ...
12, 20, 28, ...	ERAI_x_y_SPAN	Spans for AI2, AI3, ...

ERM_MA – ControlWave MICRO I/O Expansion Rack - Mixed Analog (6 AI / 2 AO) Board

DRIVER_NAME	'ERM_MA'
DATA_TYPE	DWORD (32 bits)
DRIVER_PAR1	slot number.
DRIVER_PAR2	First two bytes of Primary IP address

DRIVER_PAR3 Lower two bytes of Primary IP address

Input Map: Max Size: 40 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERAIAO_x_DRIVERSTATUS	Board Status. See Common Status Information section.
2 (bit 2)	ERAIAO_x_CALIBRATE	Bit 2 is set if the calibration data is invalid.
2 (bit 3)	ERAIAO_x_TIMEOUT	Bit 3 is set to indicate that a board data read has timed out.
4	ERAIAO_x_y_OUTRANGE	1 bit per AI, AI0 is bit 0, AI2 is bit 1. If set, output is Out-of-range.
6	ERAIAO_x_y_O_OUTRANGE	1 bit per AO AO1 is bit 0, AO2 is bit 1. If set, output is Out-of-range.
8	ERAIAO_x_y	Value for AI1 in engineering units (4-byte float - REAL). To access the value, define the variable AT %IDxx. Direct access to %IDxx is not possible.
12, 16, ...	ERAIAO_x_y	Value for AI2, AI3, ...
32,36	ERAIAO_x_y_O_ACTUAL	Value actually written to AO – clamped to 0-100%.

Output Map: Max Size: 72 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERAIAO_x_y_ZERO	Zero for AI1 (4-byte float - REAL). To access the value, define the variable AT %QDxx. This variable can be initialized at declaration.
4	ERAIAO_x_y_SPAN	Span for AI1 (4-byte float). If zero, the AI will be scaled from 0 to 100.0.
8, 16, 24, ...	ERAIAO_x_y_ZERO	Zeros for AI2, AI3, ...
12, 20, 28, ...	ERAIAO_x_y_SPAN	Spans for AI2, AI3, ...
48, 60	ERAIAO_x_y_O_ZERO	Zeros for AO1, AO2
52, 64	ERAIAO_x_y_O_SPAN	Spans for AO1, AO2
56, 68	ERAIAO_x_y_O	Values for AO1, AO2

ERM_MIX – ControlWave MICRO I/O Expansion Rack - Mixed I/O board

This card contains the following I/O: DIO 6, AI4, AO1, and HSC2

DRIVER_NAME 'ERM_MIX'

DATA_TYPE DWORD

DRIVER_PAR1 slot number.

DRIVER_PAR2 First two bytes of Primary IP address

DRIVER_PAR3 Lower two bytes of Primary IP address

Input Map: Max Size: 68 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERMIX_x_DRIVERSTATUS	Board Status. See Common Status Information section.
2 (bit 2)	ERMIX_x_CALIBRATE	Bit 2 is set if the calibration data is invalid
2 (bit 3)	ERMIX_x_TIMEOUT	Bit 3 is set to indicate that a board data read has timed out.
4	ERMIX_x_y_AI_OUTRANGE	AI under / over range. One bit per point: AI1 in bit 0, AI4 in bit 3.
6	ERMIX_x_y_AO_OUTRANGE	AO under / over range. AO1 is in bit 0.
8	ERMIX_x_y_DI	Current status of DI1 (in bit 0) to DI6 (in bit 5).
10	ERMIX_x_y_DO_I	Read-back of current value for DO1 (in bit 0) to DO6 (in bit 5).
12	ERMIX_x_TIMESTAMP	Timestamp of last sample from HSC. This is the number of Milliseconds since boot.
16	ERMIX_x_y_COUNTER	Number of counts since boot (Channel 1)
20	ERMIX_x_y_COUNTER	Counts for Channel 2
32	ERMIX_x_y_AI	Value for AI1 in engineering units (4-byte float - REAL). To access the value, define the variable AT %IDxx. Direct access to %IDxx is not possible.
36, 40, ...	ERMIX_x_y_AI	Value for AI2, AI3, ...
64	ERMIX_x_y_AO_ACTUAL	Value actually written to AO – clamped to 0-100%.

Output Map: Size: 84 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERMIX_x_y_DO	Outputs. 1 bit per value. DO1 is LSB; DO6 is MSB. Typically specified as %QXy.z, where y is I/O space offset, and z is bit number from 0 to 5.
2	ERMIX_x_y_RESET_COUNT	Counter reset flags, HSC1 (in Bit 0) to HSC2 (in bit 1)
4	ERMIX_x_LEDSTATUS	Bit 0 - If set, the diagnostic LEDs for the points are turned off to save power. Bit 1 – If set, maintains counts across Warm Start
8	ERMIX_x_y_AI_ZERO	Zero for AI1 (4-byte float - REAL). To access the value, define the variable AT %QDxx. This variable can be initialized at declaration.
12	ERMIX_x_y_AI_SPAN	Span for AI1 (4-byte float). If zero, the AI will be scaled from 0 to 100.0.
16, 24, 32	ERMIX_x_y_AI_ZERO	Zeros for AI2, AI3, AI4
20, 28, 36	ERMIX_x_y_AI_SPAN	Spans for AI2, AI3, AI4

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
72	ERMIX_x_y_AO_ZERO	Zero for AO1 (4-byte float - REAL). To access the value, define the variable AT %QDxx. This variable can be initialized at declaration.
76	ERMIX_x_y_AO_SPAN	Span for AO1 (4-byte float). If zero, the AO will be scaled from 0 to 100.0.
80	ERMIX_x_y_AO	Value for AO1 (4-byte float).

ERM_HSC4 – ControlWave MICRO I/O Expansion Rack - 4 Channel High Speed Counter Board

DRIVER_NAME	'ERM_HSC4'
DATA_TYPE	DWORD (32 bits)
DRIVER_PAR1	slot number.
DRIVER_PAR2	First two bytes of Primary IP address
DRIVER_PAR3	Lower two bytes of Primary IP address
Input Map:	Max Size: 24 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERHSC_x_DRIVERSTATUS	Board Status. See Common Status Information section.
4	ERHSC_x_TIMESTAMP	Timestamp of last sample from HSC. This is the number of milliseconds since boot.
8	ERHSC_x_y_COUNTER	Number of counts since boot (Channel 1)
12, 16, 20	ERHSC_x_y_COUNTER	Counts for Channel 2, 3, 4

Output Map: Size: 8 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERHSC_x_y_RESET_COUNT	Counter reset flags, HSC1 (in bit 0) to HSC4 (in bit 3)
2 (bit 1)	ERHSC_x_NOINIT	Maintain Counts across Warm Start (TRUE = maintain)
4	ERHSC_x_LEDSTATUS	Single bit. If bit is set, the diagnostic LEDs for the points are turned off to save power.

ERM_STAT – ControlWave MICRO I/O Expansion Rack Status Board

In addition, an Expansion Rack Status board (ERM_STAT) is defined. This board has a similar I/O map as the status board for the ControlWave Expansion Rack (ER_STAT).

DRIVER_NAME	'ERM_STAT'
DATA_TYPE	DWORD (32 bits)
DRIVER_PAR1	slot number (ignored – specify as zero).
DRIVER_PAR2	First two bytes of Primary IP address
DRIVER_PAR3	Lower two bytes of Primary IP address
Input Map:	Max Size: 1242 bytes

Note: Data starting at offset 12 will only be refreshed as fast as a 1 second interval.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERSTAT_x_BOARDSTATUS	Board Status. See Common Status Information section.
4 (bit 0)	ERSTAT_x_BATSTAT	Memory battery status at Expansion Rack. TRUE indicates good battery.
4 (bit 1)	ERSTAT_x_HOTCARDSTAT	TRUE indicates that a HOT Card replacement is in progress at the Expansion Rack. Will always be FALSE on platforms which do not support on-line HOT Card replacement.
8	ERSTAT_x_HOTCARDCT	32-Bit Count of Hot Card replacement events which have occurred.
12	ERSTAT_x_DOWNTIMEUSER	32 Bit Count – Number of seconds (as configured by the user) that the Expansion Rack can be powered off before the outputs are reset to defaults when the unit is powered back up.
16	ERSTAT_x_DOWNTIMEACT	32-Bit Count – Number of seconds that the Expansion Rack was powered off on the last power fail.
20	ERSTAT_x_WRITECT	32-Bit Count – Number of I/O updates sent to the Expansion Rack
24	ERSTAT_x_READCT	32-Bit Count – Number of I/O updates sent from the Expansion Rack
28	ERSTAT_x_CONNECTS	32-Bit Count – Number of Connects and Disconnects made to the Expansion Rack.
32	ERSTAT_x_HEARTBEAT	32-Bit Count – Number of IDLE time Heartbeats sent from the Expansion Rack to the host.
42	ERSTAT_x_BDSTR1	String – I/O board string for slot #1
127	ERSTAT_x_BDSTR2	String – I/O board string for slot #2
212	ERSTAT_x_BDSTR3	String – I/O board string for slot #3
297	ERSTAT_x_BDSTR4	String – I/O board string for slot #4
382	ERSTAT_x_BDSTR5	String – I/O board string for slot #5
467	ERSTAT_x_BDSTR6	String – I/O board string for slot #6
552	ERSTAT_x_BDSTR7	String – I/O board string for slot #7
637	ERSTAT_x_BDSTR8	String – I/O board string for slot #8
724		DINT – Current redundancy status of the Expansion Rack – will always be zero on this platform

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
728	ERSTAT_x_BDSTR9	String – I/O board string for slot #9
813	ERSTAT_x_BDSTR10	String – I/O board string for slot #10
898	ERSTAT_x_BDSTR11	String – I/O board string for slot #11
983	ERSTAT_x_BDSTR12	String – I/O board string for slot #12
1068	ERSTAT_x_BDSTR13	String – I/O board string for slot #13
1153	ERSTAT_x_BDSTR14	String – I/O board string for slot #14
1240	ERSTAT_x_INPUTVOLTS	Float – Reading (in volts) of power-supply input voltage.

Output Map: Max Size: 4 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0.0		If this is a Redundant Expansion Rack, and the Standby is valid, then writing a TRUE to this location will cause the rack to fail-over. Not used on this platform.

ERM_TC6 – ControlWave MICRO I/O Expansion Rack 6 Point Thermocouple Board

DRIVER_NAME 'ERM_TC6'
 DATA_TYPE DWORD (32 bits)
 DRIVER_PAR1 slot number.
 DRIVER_PAR2 First two bytes of Primary IP address
 DRIVER_PAR3 Lower two bytes of Primary IP address
 Input Map: Max Size: 32 bytes

Due to the amount of time required to process the thermocouple points, it is highly recommended that this I/O driver be assigned to a task (instead of “No Task”). Care should also be taken in using the I/O board in a task of less than 40ms.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	TC6_x_DRIVERSTATUS	Board Status. See Common Status Information section.
2 (Bit 2)	TC6_x_CALIBRATE	If set indicates invalid calibration data written to the board.
2 (Bit 3)	TC6_x_TIMEOUT	If set, indicates that had an error reading or writing to the board.
4	TC6_x_y_OUTRANGE	1 bit per TC, TC1 is bit 0, TC6 is bit 5. If set, input is Out-of-range.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
8	TC6_x_y	Value for TC1 in engineering units (4-byte float - REAL). To access the value, define the variable AT %IDxx. Direct access to %IDxx is not possible.
12, 16, 20, 24, 28	TC6_x_y	Value for TC2, TC3, TC4, TC5, TC6

Output Map: Max Size: 70 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	TC6_x_y_ZERO	Zero for TC1 (4-byte float - REAL). To access the value, define the variable AT %QDxx. This variable can be initialized at declaration.
4	TC6_x_y_SPAN	Span for TC1 (4-byte float). If zero, the TC will be scaled as in the chart below. If specified, the new value will be $ORG_VALUE * Span + Zero$.
8, 16, 24, 32, 40	TC6_x_y_ZERO	Zeros for TC2, TC3, TC4, to TC6 – Example: for C to F, use 32.0
12, 20, 28, 36, 44	TC6_x_y_SPAN	Spans for TC2, TC3, TC4, to TC6 – Example for C to F, use 1.8
64	TC6_x_y_MODE	Point type for TC1; see Thermocouple type codes section for details.
65, 66, 67, 68, 69	TC6_x_y_MODE	Point types for TC2, TC3, TC4, TC5, and TC6.

Type codes for Thermocouple Points.

Type Code	Code	Range
0	B	Thermocouple: 100C – +1820C
1	E	Thermocouple: -270C – +1000C
2	J	Thermocouple: -210C – +1200C
3	K	Thermocouple: -270C – +1370C
4	R	Thermocouple: -50C – +1720C
5	S	Thermocouple: -50C – +1760C
6	T	Thermocouple: -270C – +400C
7	Unused	Unused
8	10MV	Voltage Inputs: -10 mV to +10 mV (Outputs as 0.0 to 1.0)
9	C	Thermocouple: 0C – +2315C
10	N	Thermocouple: -270C – +1300C

ERM_RTD4 – ControlWave Micro I/O Expansion Rack – 4 Point Resistance Temperature Device (RTD) Board

DRIVER_NAME	'ERM_RTD4'
DATA_TYPE	DWORD (32 bits)
DRIVER_PAR1	slot number.
DRIVER_PAR2	First two bytes of Primary IP address
DRIVER_PAR3	Lower two bytes of Primary IP address
Input Map:	Max Size: 24 bytes

Due to the amount of time required to process the RTD points, it is highly recommended that this I/O driver be assigned to a task (instead of “No Task”). Care should also be taken in using the I/O board in a task of less than 40ms.

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERRTD_x_DRIVERSTATUS	Board Status. See Common Status Information section.
2 (Bit 2)	ERRTD_x_CALIBRATE	If set indicates invalid calibration data written to the board.
2 (Bit 3)	ERRTD_x_TIMEOUT	If set, indicates that had an error reading or writing to the board.
3 (Bit 0)	ERRTD_x_LASTCALBOP	Set if last calibration or reset operation failed.
3 (Bit 7)	ERRTD_x_CALBCMD	Calibration Commands Allowed. Until this bit is set, all calibration commands are ignored.
4	ERRTD_x_y_READERR	RTD Reading Error. Bit 0 is RTD1, Bit 3 is RTD4
8	ERRTD_x_y	RTD1 reading – REAL – In units of Degrees Centigrade (unless scaled by values in the output map).
12, 16, ...20	ERRTD_x_y	Readings for RTD2 ... RTD4.

Output Map: Max Size: 200 bytes

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
0	ERRTD_x_y_ZERO	Zero for RTD 1(4-byte float - REAL). Example: for C to F, use 32.0. Defaults to 0.0
4	ERRTD_x_y_SPAN	Span for RTD 1(4-byte float – REAL). If zero, RTD will not be scaled. If specified, the scaled value will be $ORG_VALUE * Span + Zero$. Example for C to F, use 1.8.
8,16,24,32 40,48,56	ERRTD_x_y_ZERO	Zero for RTDs 2-8
12,20,28, 36,44,52, 60	ERRTD_x_y_SPAN	Span for RTDs 2-8

Offset	Default Variable Name where x is the board slot and y is the pin number.	Description
64	ERRTD_x_y_MODE	
65,66,67, 68,69	ERRTD_x_y_MODE	
100		NOT USED; RESERVED FOR FUTURE USE
120 (Bit 0) *	ERRTD_x_y_RESTORE	If set, restore RTD 1 calibration to Factory Defaults. Will be reset when operation completes.
121 *	ERRTD_x_y_OPERATION	Calibration Operation for RTD 1 – SINT 5 RTD Zero (100 Ohms) 6 RTD Span (300 Ohms) 7 RTD Coefficients (A, B, R0) 8 RTD Span (not using 300 Ohms)
124**	ERRTD_x_y_COEFF_A	Coefficient A (RTD 1)
128**	ERRTD_x_y_COEFF_B	Coefficient B (RTD 1)
132**	ERRTD_x_y_COEFF_R0	Coefficient R0 (RTD 1)
136**	ERRTD_x_y_APPLIED	The applied temperature when calibration operation 8 was performed. (RTD 1)
140 (Bit 0) *	ERRTD_x_y_RESTORE	If set, restore RTD 2 calibration to Factory Defaults. Will be reset when operation completes.
141 *	ERRTD_x_y_OPERATION	Calibration Operation for RTD 2 – SINT 5 RTD Zero (100 Ohms) 6 RTD Span (300 Ohms) 7 RTD Coefficients (A, B, R0) 8 RTD Span (not using 300 Ohms)
144**	ERRTD_x_y_COEFF_A	Coefficient A (RTD 2)
148**	ERRTD_x_y_COEFF_B	Coefficient B (RTD 2)
152*	ERRTD_x_y_COEFF_R0	Coefficient R0 (RTD 2)
156*	ERRTD_x_y_APPLIED	The applied temperature when calibration operation 8 was performed. (RTD 2)
....	
180 (Bit 0) *	ERRTD_x_y_RESTORE	If set, restore RTD 4 calibration to Factory Defaults. Will be reset when operation completes.
181 *	ERRTD_x_y_OPERATION	Calibration Operation for RTD 4 – SINT 5 RTD Zero (100 Ohms) 6 RTD Span (300 Ohms) 7 RTD Coefficients (A, B, R0) 8 RTD Span (not using 300 Ohms)
184**	ERRTD_x_y_COEFF_A	Coefficient A (RTD 4)
188**	ERRTD_x_y_COEFF_B	Coefficient B (RTD 4)
192**	ERRTD_x_y_COEFF_R0	Coefficient R0 (RTD4)
196**	ERRTD_x_y_APPLIED	The applied temperature when calibration operation 8 was performed. (RTD 4)

* Value written to perform operation. The value will be reset by driver when the operation completes.

** Value is read from the board by the driver. In order to perform calibration operations 7 and 8, the user can overwrite the values; then, issue the calibration command.

I/O Simulator

What is the I/O Simulator?

The control programs generated through ControlWave Designer are executed by the ControlWave controller using the IEC 61131 real time system, working in conjunction with ControlWave firmware.

The PC-based **I/O Simulator** operates using copies of the IEC 61131 real time system and ControlWave firmware which are identical to those running in the controller. This allows any control strategy generated for ControlWave Designer to be tested on a PC, with simulated analog and digital inputs and outputs. Initial I/O testing and debugging may be performed in a safe, isolated environment, without the need for a running ControlWave controller and process I/O boards.

Important

The I/O Simulator is designed to work with IPCxx RTU resources. It does NOT work with ARM-based RTU resources. Therefore, projects created to run in the ControlWave MICRO-series of controllers **will not** execute in the I/O Simulator.

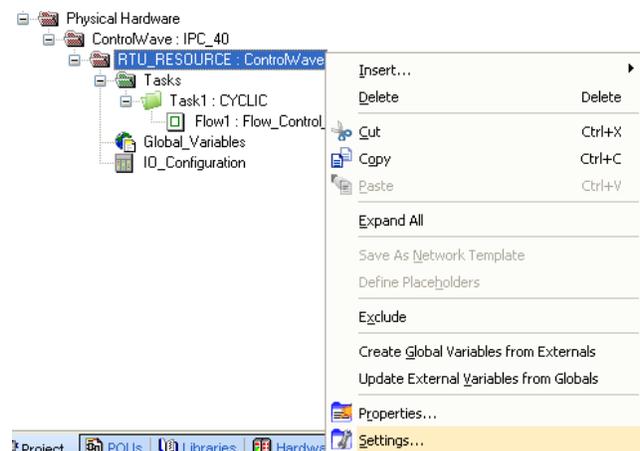
Number of Boards Available within the I/O Simulator

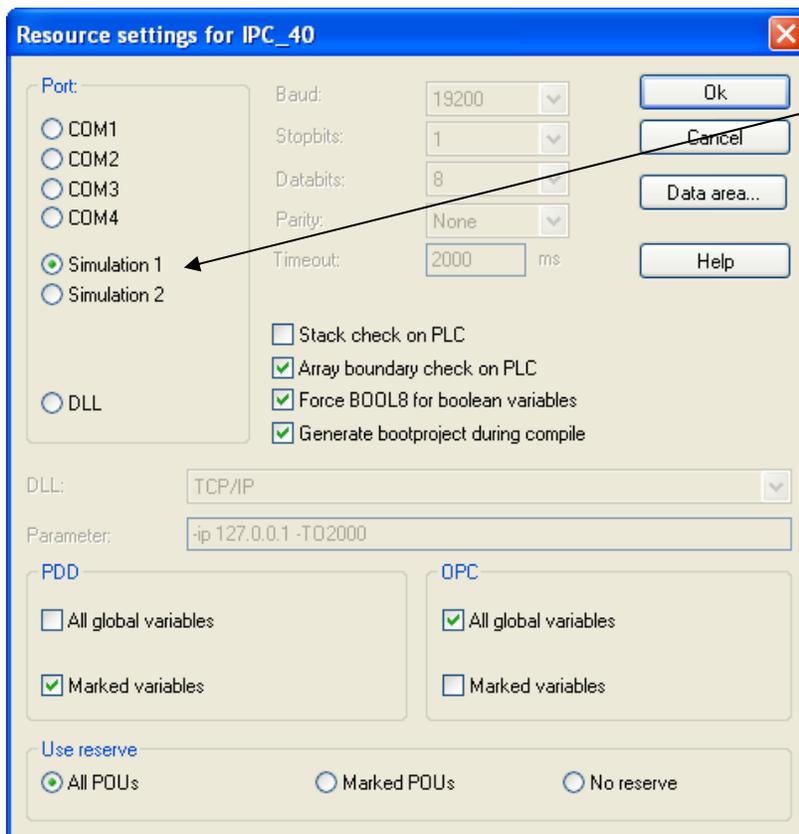
The I/O Simulator has a limit on the number of boards that may be accessible within the simulation. This limit is dictated by an internal limit of 500 characters for the list used to describe the boards used in the I/O Simulator. If, while using the I/O Simulator, some boards do not appear, you must re-run the I/O Configurator, and select for I/O simulation the group of boards that are missing and de-select unused boards, as needed, so the limit is not exceeded.

Starting the I/O Simulator

Before you can start the I/O Simulator, you must identify it as the download destination for your control program. This is done in the Resource Settings dialog box.

To choose the resource, *right* click on the resource and choose **“Settings”** from the pop-up menu.



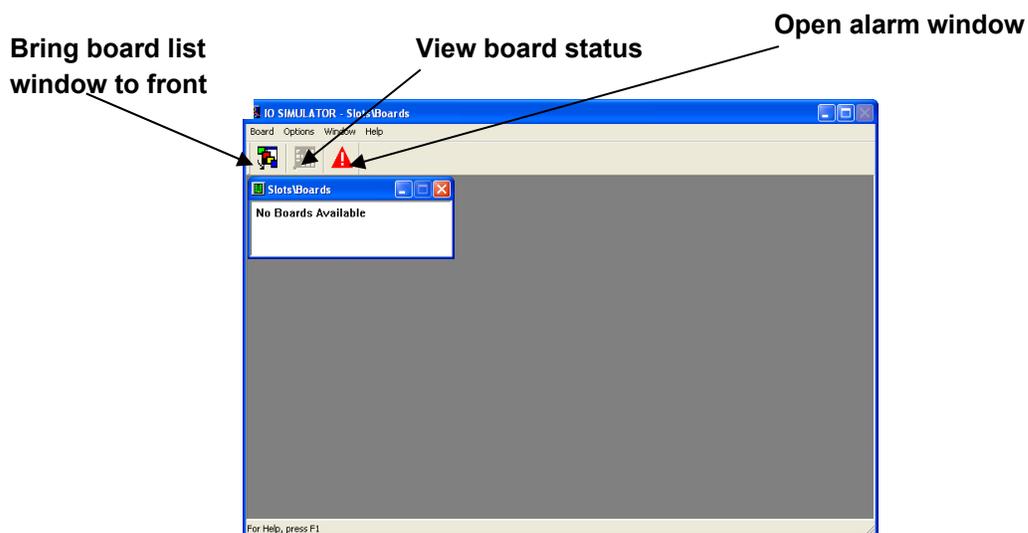


Choose "Simulation 1"

Choose "Simulation 1" in the Resource Settings dialog box, and click on [OK].

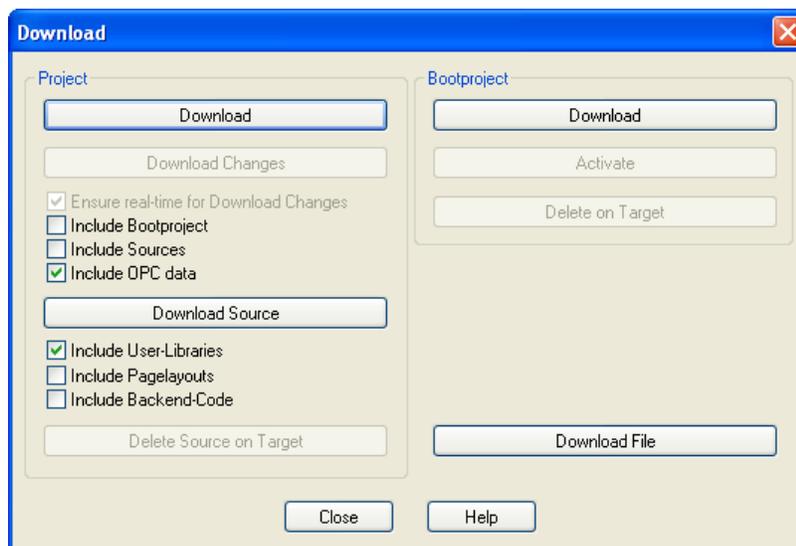
Next, click as follows: **Online**→**Project Control**

The I/O Simulator will appear, however, no I/O boards will be displayed, yet.



Minimize the I/O Simulator in order to uncover the RTU Resource dialog box.

Click on the **[Download]** button in the RTU Resource dialog box.



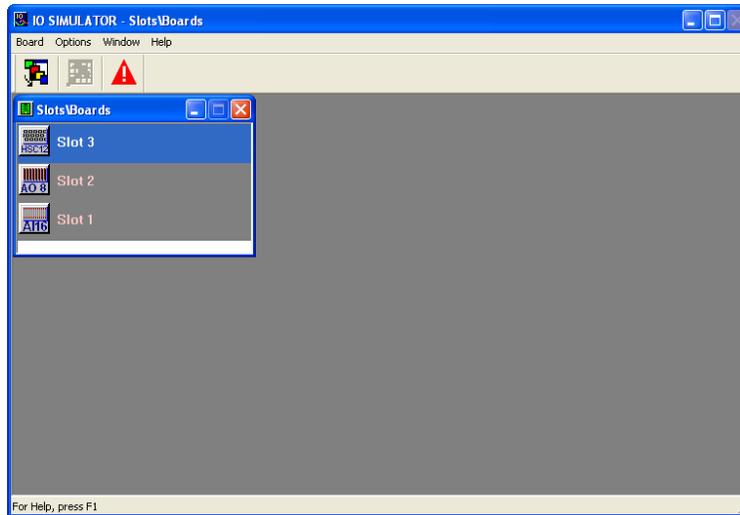
The Download dialog box will appear. Click the **[Download]** button in the Project section of the Download dialog box.

The RTU_Resource dialog box will re-appear. Click either the **[Warm]** or **[Cold]** buttons. The **[Warm]** button only re-initializes non-retentive variables, i.e. variables which are NOT marked as 'RETAIN'. The **[Cold]** button re-initializes all variables.

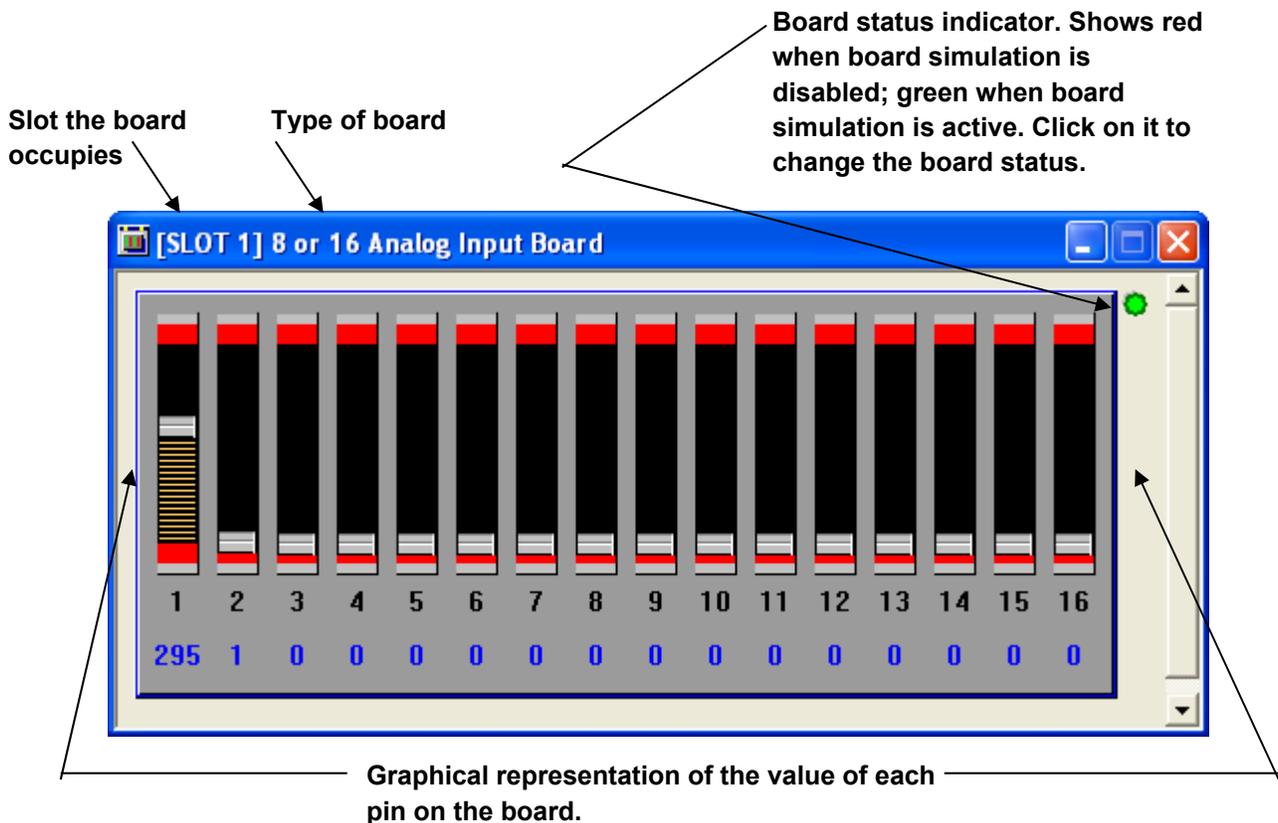
If desired, execution can be stopped by clicking the **[Stop]** button. Then, execution can be re-started using either the **[Warm]** or **[Cold]** buttons.



In the I/O Simulator, icons will now appear for all configured process I/O boards. In this example, we have two process I/O boards – one analog input board, and one analog output board. In the figure, below, they are shown as Slot 1 and Slot 2, respectively.



Click on either of the board icons, or press the [Enter] key while one of the boards is highlighted, and a graphical representation of the board, showing simulated values for all pins on the board will appear. Analog values are shown as bar graphs; digital values are shown as buttons.



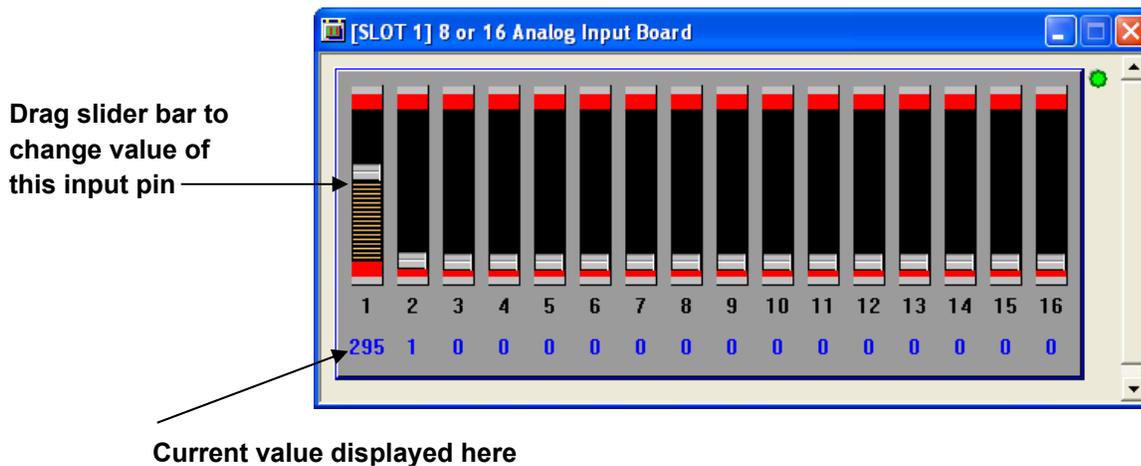
Enabling / Disabling the Board Simulation

For all board types, the board simulation can be enabled/disabled by clicking on the board status indicator in the upper right hand corner of the board graphic. This indicator will appear in green when the simulation is active, or red when the board simulation is turned off or the board has not been configured.

Analog Boards

Analog Input Boards

In analog input boards, the value of a particular pin may be altered by dragging the slider bar associated with the graphic for each pin. Alternatively, right click on the pin and choose “**Configure Pin**” from the pop-up menu. Enter a value in the “**Current Value**” field.



Analog Output Boards

Within the I/O Simulator Analog output boards are depicted similarly to analog input boards, however, you are NOT allowed to alter the value of individual pins.

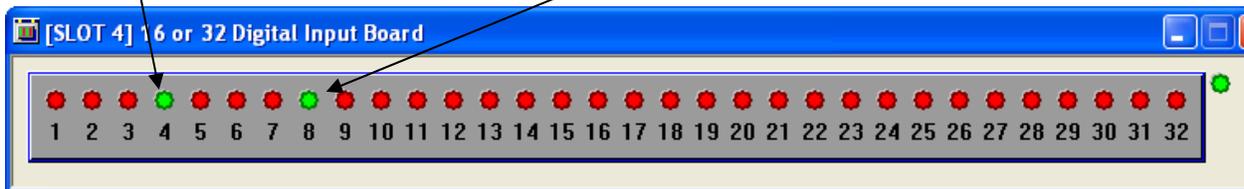
Digital Boards

Digital Input Boards

For digital input boards, the value of a particular pin may be altered by clicking on the button which corresponds to that pin. A pin is ON when its button is displayed in green; a pin is OFF when its button is displayed in red.

Pin status is displayed based on color; green is ON and red is OFF.

The pin's ON/OFF status can be toggled by clicking on it.

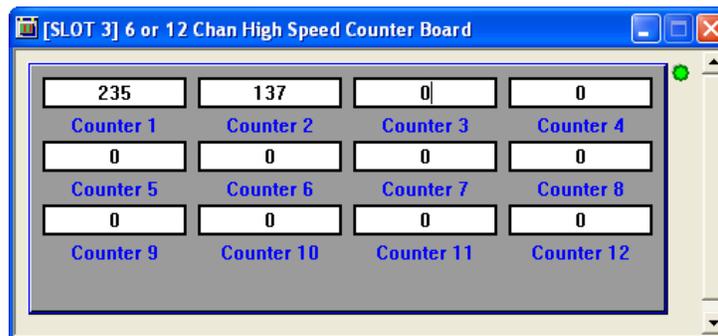


Digital Output Boards

Within the I/O Simulator, digital output boards are depicted similarly to digital input boards, however, you are NOT allowed to alter the status of individual pins.

Counter Boards

High Speed Counter input values can be displayed within the I/O Simulator. You can also enter new values for the counter inputs.



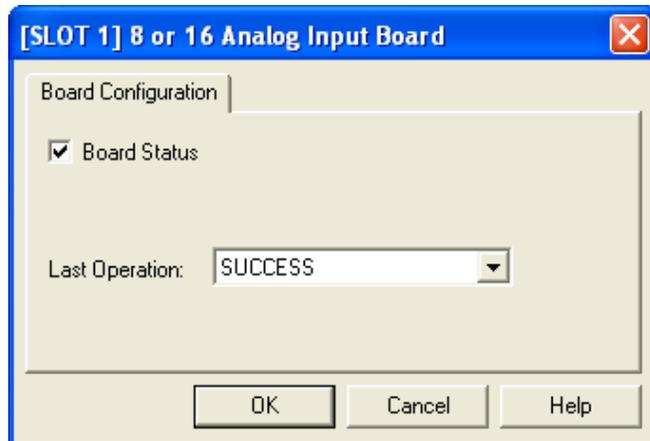
Viewing the Board Configuration Status

To view the status of a board in the I/O simulation, click on **Options**→**Configure Current Board** or *right-click* in certain portions of the graphical representation of the board and choose **“Configure Board”** from the pop-up menu.

Board Configuration Page

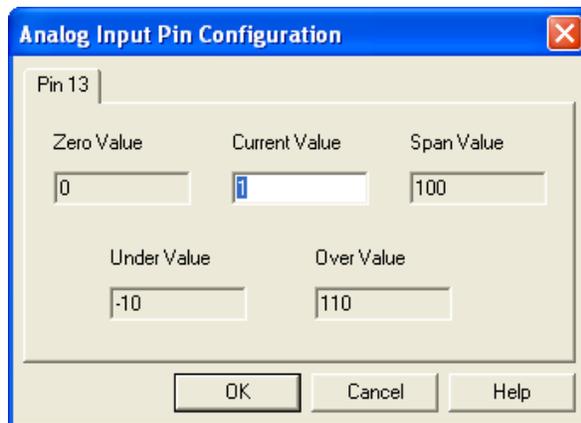
The contents of the Board Configuration page vary, depending upon what type of board you are configuring.

In all cases, you can disable the simulation for the board, by de-selecting the **“Board Status”** check box; this causes the text ‘NOT PRESENT’ to appear. This is equivalent to setting the board status indicator to OFF.



Configuring a Pin

For boards which simulate input values, you can change the value / status of a particular pin, by right-clicking on the graphical representation for that I/O point, and choosing **Configure Pin** from the pop-up menu. You can then enter a new value (for analog boards) or toggle the status (for digital boards). Output board types only allow you to view information about the pin, not to change it.





Viewing Simulated Alarms

As you proceed to manipulate process variables in the I/O simulator, you can generate alarm messages, if those process variables are connected to alarm function blocks. The alarm messages can then be viewed in the Alarm Window. To view alarm messages in the I/O Simulator's Alarm Window, click on the 'Alarm Window' icon, or click on **Options**→**Display Alarm Window**.

Alarm Window							
Wed Sep 13	15:47:58	2000->	MYPROG1.F105_LEVEL	HiHi	Single		87
Wed Sep 13	15:46:38	2000->	MYPROG1.F105_LEVEL	HiHi	Single	Critical	98
Wed Sep 13	15:45:32	2000->	MYPROG1.F105_LEVEL	LoLo	Single		75
Wed Sep 13	15:45:17	2000->	MYPROG1.F105_LEVEL	LoLo	Single	Event	0

Shutting Down the I/O Simulator

The I/O Simulator cannot be shutdown from a board window; it will re-appear if you attempt this. The only way to shut it down is by choosing the **[Close]** button from the RTU_Resource dialog box.

Note:

If the RTU_Resource dialog box is not visible, minimize other windows of the I/O Simulator to uncover it.

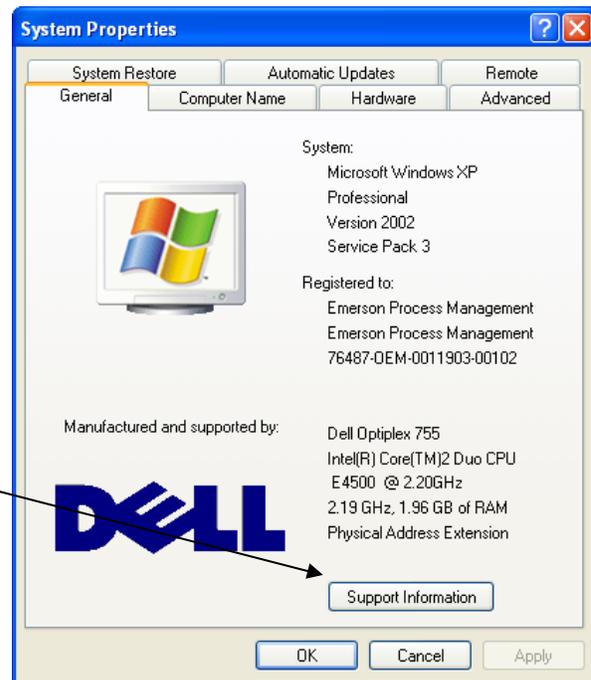
Troubleshooting Tip

For some Windows operating systems, if **Physical Address Extension** capability is turned ON, it *may* prevent the I/O Simulator from running and generate an error. If you are having trouble running the simulator you should check to see your operating system supports this capability, and if it does, try turning it OFF and see if this remedies the situation.

To see if Physical Address Extension is supported:

Double-click on **System** in the Windows Control Panel. If you see **Physical Address Extension** displayed on the **"General"** tab, it means your operating system supports this.

If you see “Physical Address Extension” displayed here, it means your operating system supports it.



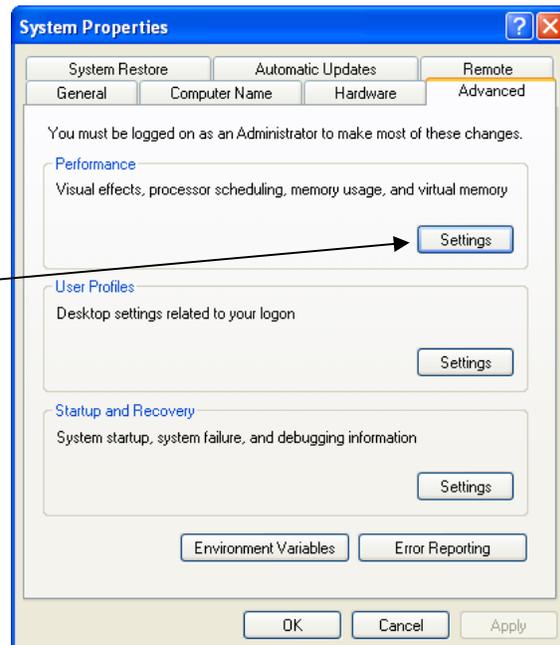
To Turn Off Physical Address Extension:

There are two methods available for turning OFF physical address extension capability.

Method 1:

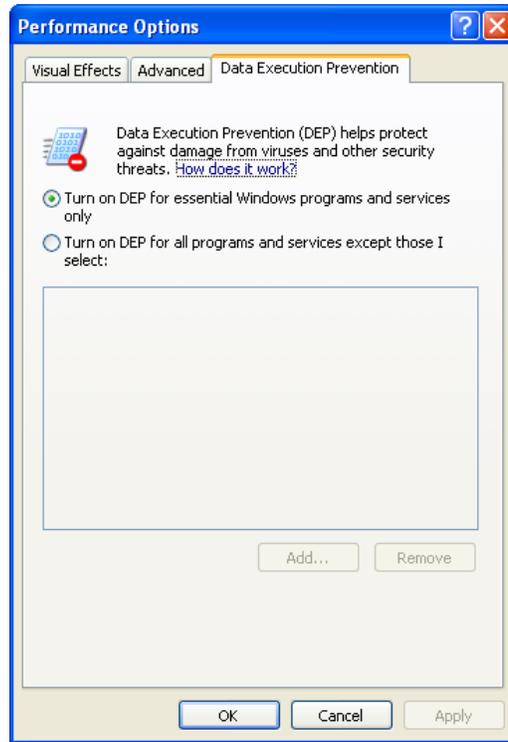
1. In the System Properties dialog box, click the **Advanced** tab.
2. Click **Settings** in the Performance section.

Click “Settings”.



3. In the Performance Options dialog box, click the **Data Execution Prevention** tab.

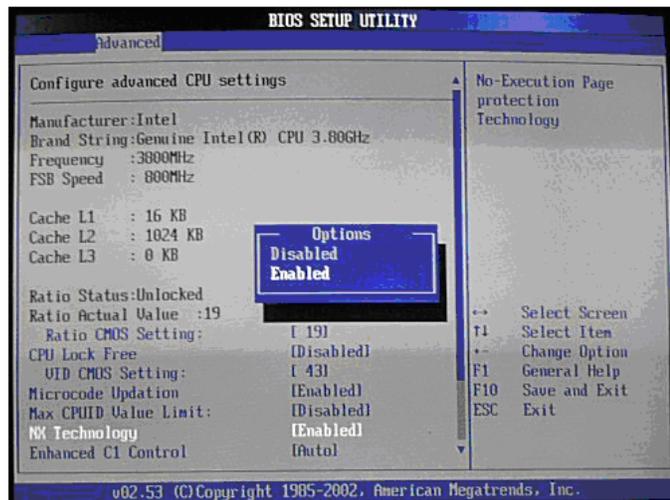
- On the Data Execution Prevention tab, check the button for **Turn on DEP for essential Windows programs and services only**.



- Click **OK** and reboot your PC for the change in settings to take effect.

Method 2:

If you are familiar with configuring system BIOS, you can turn physical address extension OFF in the BIOS by disabling NX technology.



IP Addressing and Networks

Internet Protocol (IP) is one method in which the ControlWave controller can communicate on a network. IP is supported through the ControlWave's Ethernet port(s), and serial IP (PPP) is supported through its serial ports.

What is the Format of IP Addresses?

Each network connection from an IP node has an **IP address** which is unique within the network. It is important to note that the IP address is associated with the *network connection* (IP Port) on the node, NOT the node itself. This allows a single IP node to have *more than one* port, and consequently, more than one IP address.

IP addresses consist of 32 **bits** (1's and 0's) which are divided up into 4 groups of 8 bits each. A period is used to separate each group. Each group of 8 bits is then converted from binary to a decimal number from 0 to 255. The resulting IP address is said to be in **dotted decimal** notation.

Each group of 8 bits is converted to a decimal number

01111000 . 00000000 . 11010010 . 00000001
↓ ↓ ↓ ↓
120 . 0 . 210 . 1

Each of the numbers in the address generally has a specific meaning. The IP address is generally divided up into a **network portion** which must be common to each node in the network, and a **local portion** of which some part *must be unique* to a particular node.

How is the Specific Meaning of Each Part of the Address Defined?

Addresses must be assigned to be consistent with whatever conventions have been established for your system. For example, if this network has connections outside the plant (such as connection to the real world-wide Internet), then the choice of this network number is assigned by an Internet governing body called the Network Information Center (NIC) or whatever Internet service provider you are using. In addition, there are certain rules to defining addresses, which will be discussed later.

The specific meaning of each part of the address is defined in something called the IP mask or **sub-net mask**. The sub-net mask is simply another set of 32 bits (which must also be converted to dotted decimal notation). Each bit in the sub-net mask corresponds to a bit in the IP address. If a bit in the sub-net mask is set to 1 (ON), then the corresponding bit *in the IP address* is considered to be part of the **network portion** of the IP address. The network portion can be ignored (or '*masked*') when performing communications to *nodes within the same network*, because by definition, all nodes in the same network have identical network

portions. Any bit in the sub-net mask which is 0 (OFF) is considered to be part of the local addressing scheme.

The figure, below, shows the IP address and corresponding sub-net mask for an IP address of 120.0.210.1 and a sub-net mask of 255.0.0.0.

32-bit IP Address: (dotted decimal 120.0.210.1)

01111000 . 00000000 . 11010010 . 00000001

32-bit Sub-net Mask: (dotted decimal 255.0.0.0)

11111111 . 00000000 . 00000000 . 00000000

All 1's in the Sub-net Mask indicate that the corresponding bits in the IP Address are used for the Network portion of the address.

All 0's in the Sub-net Mask indicate that the corresponding bits in the IP Address are used for whatever local addressing scheme has been defined.

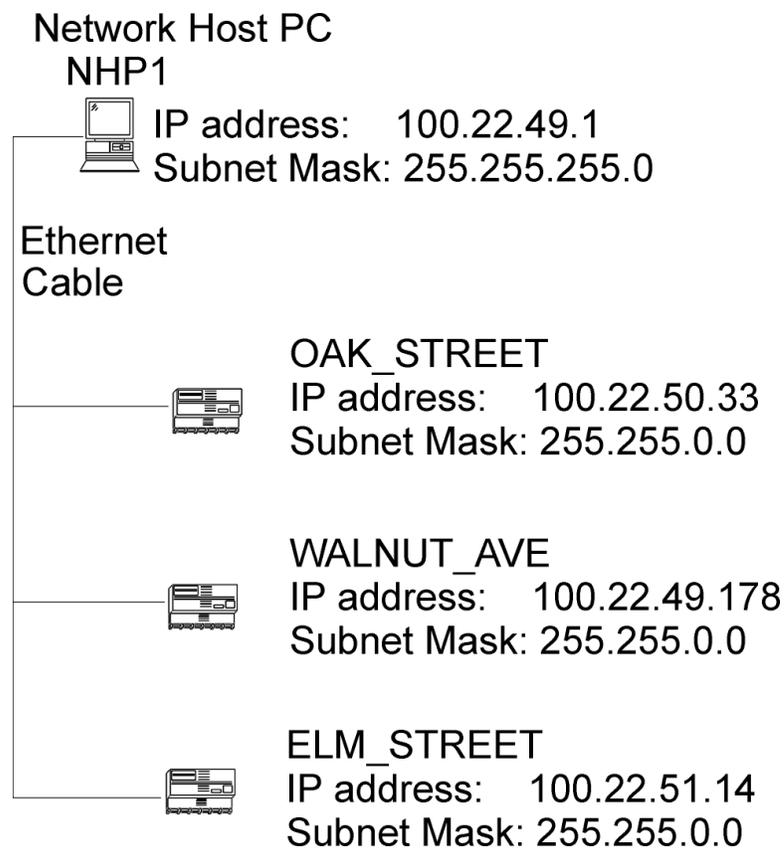
As we said before, a '1' in the sub-net mask indicates that the corresponding IP address bit is part of the network portion of the address. Because the first part of the IP address '01111000.' has a corresponding sub-net mask of '11111111' we know that '01111000' (120 in decimal) is the network portion of the address.

The remaining parts of the IP address '00000000.11010010.00000001' have a corresponding sub-net mask of '00000000.00000000.00000000'. These bits are used as part of the local communications addressing scheme.

A node's IP address, and its sub-net mask, define the range of acceptable addresses with which the node can communicate. For example, if one node has an IP address of 4.3.2.1 and another node has an IP address of 100.100.0.1, there is no common network portion between the two addresses. For that reason, there is NO way these two nodes can communicate with each other directly - - they are each part of *different* networks. Any messages between these nodes would have to pass through one or more **router** computers.

For two nodes to communicate directly, the network portion of their addresses (as specified by the sub-net mask) must match exactly.

To illustrate this concept, look at the figure, below. The network shown has one Network Host PC (NHP) called NHP1, and 3 controllers (RTUs) named OAK_STREET, ELM_STREET, AND WALNUT_AVE.



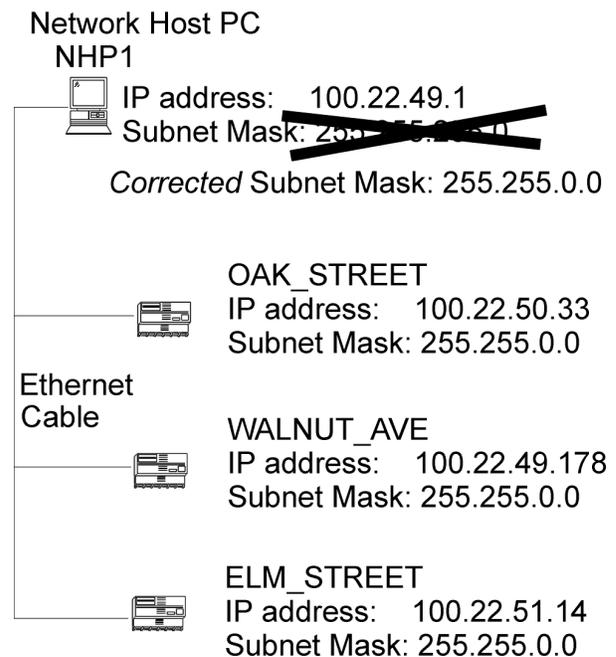
However, the table below reveals a problem with the configured sub-net masks.

Node Name	IP Address, Subnet Mask:	Mask Says This Node Can Send Messages to All Nodes with Addresses:
NHP1	IP ADR: 100.22.49.1 MASK: 255.255.255.0	100.22.49.yyy where yyy is an integer from 0 to 255.
WALNUT_AVE	IP ADR: 100.22.49.178 MASK: 255.255.0.0	100.22.yyy.zzz where yyy and zzz are integers from 0 to 255.
OAK_STREET	IP ADR: 100.22.50.33 MASK: 255.255.0.0	100.22.yyy.zzz where yyy and zzz are integers from 0 to 255.
ELM_STREET	IP ADR: 100.22.51.14 MASK: 255.255.0.0	100.22.yyy.zzz where yyy and zzz are integers from 0 to 255.

Based on their specified IP addresses and sub-net masks, OAK_STREET, ELM_STREET, and WALNUT_AVE can all communicate with each other. They can also send messages to NHP1.

There is a problem, however. NHP1 has a sub-net mask which specifies that it can only send messages to nodes with addresses 100.22.49.*nnn* where *nnn* is an integer from 0 to 255. The only node which it can send messages to, therefore, is WALNUT_AVE.

To remedy this situation, NHP1's sub-net mask *should be changed to* 255.255.0.0 so that it can also send messages to OAK_STREET and ELM_STREET. The corrected sub-net mask is reflected in the figure at right.

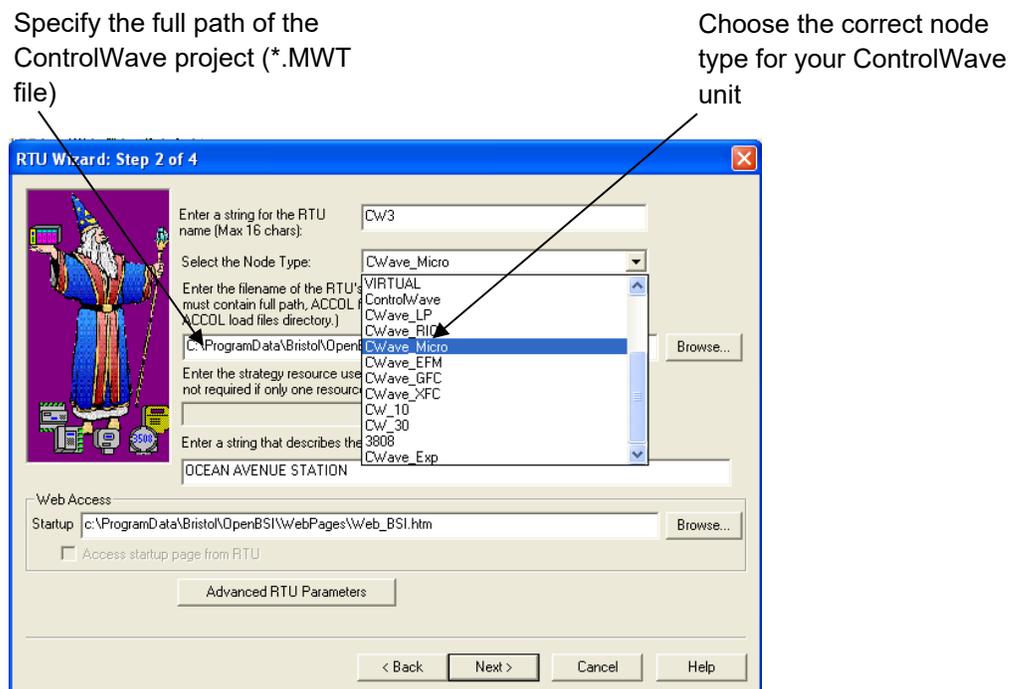


Adding a ControlWave to an IP Network with the RTU Wizard

Within OpenBSI's NetView program, the ControlWave can be added to an existing IP network in the same way as you would add any other controller.

Within the NetView tree, simply choose the icon for the network to which you want to add the ControlWave, *right-click* on the icon, and choose **Add→RTU** to call up the RTU Wizard.

In the RTU Wizard, be sure you specify its node type, for example, 'ControlWave' 'CWave_MICRO', etc. and that you also specify the full path of the ControlWave project .MWT file.



In addition, you should specify the startup web page for the controller. If the startup web page is on the PC, specify its full path. If it resides within the ControlWave, just specify the name and select the **"Access startup page from RTU"** check box.

You will also need to specify a local address for the ControlWave, and, as well as an IP address and IP mask for the ControlWave. The local address must match whatever local address has been defined for the ControlWave in the Port Configuration web page or Flash Configuration Utility 'Ports' page. The default is 1. This subject is discussed, in more detail, later.

Note:

To view web pages in OpenBSI, the path of Microsoft® Internet Explorer (IEXPLORE.EXE) must be entered in the **“Web Browser”** field of the OpenBSI Application Parameters dialog box in NetView. To access the OpenBSI Application Parameters dialog box, sign on to NetView with the system password, and then click on the Application Parameters icon. For more information on the OpenBSI Application Parameters dialog box, and the NetView program, see the *OpenBSI Utilities Manual* (part number D301414X012).

Full details on adding controllers to OpenBSI networks are included in Chapter 6 of the *OpenBSI Utilities Manual* (part number D301414X012).

Setting up IP Ports in the Flash Configuration Utility

The ‘Ports’ page in the Flash Configuration Utility allows you to configure the characteristics of the ControlWave-series controller’s serial and Ethernet ports. See the ‘*IP Ports – Ethernet*’ and ‘*IP Ports – PPP*’ sections in this manual for more information.

Recommended Ranges for IP Addresses

If you are intending to connect your OpenBSI network directly to the global world-wide Internet, you must obtain a range of IP addresses from your Internet service provider (ISP) or from an Internet governing body such as the Internet Assigned Numbers Authority (IANA).

If you have no plans to connect your network to the global Internet, there is no restriction on your choice of IP addresses, however, the Internet Engineering Task Force recommends (in accordance with *RFC 1918*, or *Rekhter, et al, Best Current Practice memo - Address Allocation for Private Internets, Internet Engineering Task Force, February, 1996*; see <http://www.ietf.org> for the complete text of this memo) that IP addresses for private networks should be assigned from the following ranges:

10.0.0.0 to 10.255.255.255

172.16.0.0 to 172.31.255.255

192.168.0.0 to 192.168.255.255

These particular ranges of Internet addresses have been set aside for private networks. Any messages coming from these addresses can be recognized by most Internet Service Providers (ISP) as coming from private networks, and so can be filtered out. This helps avoid addressing conflicts should an accidental connection occur between a private network, and the global Internet.

Devices (e.g. controllers, workstations) in networks created with OpenBSI must always use fixed IP addresses. This causes certain complexities if you choose to use Dynamic Host

Configuration Protocol (DHCP) in your network to provide addresses to other non-ControlWave or Network 3000 devices. Because DHCP assigns IP addresses dynamically, as they are needed, you must examine your DHCP server to determine the addresses which have been assigned for each ControlWave or Network 3000 controller or OpenBSI workstation and then *manually* enter those addresses in NetView. You should then specify the longest possible lease time for the addresses, to help prevent the loss of a given address through a device failure.

It is also strongly recommended that the DHCP server is configured such that the addresses reserved for the controllers are permanently reserved (by tying them to the RTU MAC addresses within the DHCP configuration or by having them in a totally different address range). The same should be done when configuring RAS servers or other machines capable of providing dynamic addressing information. Otherwise, you can easily have duplicate IP addresses on your network.

IP Parameters

The IP Parameters page is accessible from the **"IP Parameters"** tab in the Flash Configuration Utility. This page is used to specify the IP addresses (in dotted decimal format) of this controller's Network Host PC (NHP), as well as UDP port/socket information. Additional parameters are available related to IP routing, and communications security.

The screenshot shows the 'IP Parameters' configuration window. At the top, there are tabs: 'Soft Switches', 'Ports', 'IP Parameters' (active), 'Application Parameters', 'Archive', 'Audit', and 'IP Routes'. Below the tabs, the 'NHPs' section has two IP address fields: 'IP ADDR A' (10.211.75.154) and 'IP ADDR B' (0.0.0.0). The 'UDP Ports' section has 'IBP' (1234) and 'Time Sync' (1235). The 'Gateway' section has 'Default G/W' (0.0.0.0). The 'RIP Protocol' section has 'Inclusion Addr' (0.0.0.0), 'Exclusion Addr' (0.0.0.0), 'Inclusion Mask' (0.0.0.0), and 'Exclusion Mask' (0.0.0.0). The 'Dynamic IP Routing Ping' section has 'Rate' (30000 ms), 'Timeout' (15000 ms), and 'Retries' (1). The 'Challenge Protocol' section has a 'Default Username' field.

NHPs:

IP ADDR A: This is the *primary* IP address for this controller's Network Host PC (NHP). It must be entered in dotted decimal format.

IP ADDR B: This is a *secondary* IP address for the same NHP referenced by "IP ADDR A" or the IP address of a redundant backup NHP. It must be entered in dotted decimal format. If neither of these situations apply, leave **IP ADDR B** blank.

UDP Ports:

IBP: This is the UDP port number (socket number) used by the IP driver. It is used to split message traffic along different *streams*. All PCs or RTUs which are to communicate with each other must have the same **"IBP"** number. In a sense, this value is like a common password which must be known by each node in the network. If no value is entered, a default value from the NETDEF files is used. (NOTE: Although the term *UDP Port* is used, it has no actual relationship with the physical communication ports.)

Time Sync: This is the UDP port number (socket number) used for time synchronization of the RTUs. All PCs or RTUs must have this value defined, or else they will be unable to receive time synchronization messages. In a sense, this value is like

a common password which must be known by each node in the network. If no value is entered, a default value from the NETDEF files is used. (NOTE: Although the term *UDP Port* is used, it has no actual relationship with the physical communication ports.)

SNMP: (Requires CWP/LPS/CWR 03.00 or newer firmware)

Disable SNMP Processing: SNMP allows certain IP parameters to be monitored and adjusted remotely. For security purposes, you may want to check this box to disable this capability.

Other notes about SNMP:

If you choose to leave SNMP processing enabled, you should be aware of the following things:

ControlWave supports RFC1213 (MIB II).

For the community string, any string is accepted to read data. For writes (updates) the community string must be specified as a valid <username>/<password> combination.

The system contact, description, and location strings are taken from the `_CW_CONTACT_STR`, `_CW_DESCRIPTION_STR`, and `_CW_LOCATION_STR` system variables. See the *System Variables* section of this manual for more information on these strings.

Gateway:

Default G/W: This is an IP address of a default gateway. The default gateway is an address to which the system sends any messages with destinations that are not directly reachable (that is, not in the address range specified via the IP mask for this node). This address must be entered in dotted decimal format. For more information on using gateways in your network, see Chapter 1 of the *OpenBSI Utilities Manual* (part number D301414X012).

RIP Protocol:

This section allows configuration of parameters for the **Routing Internet Protocol (RIP)** (refer to *Douglas Comer and David Stevens, Internetworking with TCP/IP - Volumes 1 & 2* (Englewood Cliffs, NJ: Prentice Hall, 1991); *Frank Derfler and Steve Rigney, TCP/IP A Survival Guide for Users* (New York: MIS Press, 1998)). **RIP** is used to support **dynamic IP routing** (described below) and is implemented beginning in ControlWave firmware CWP02.0. A router which supports RIP essentially maintains a set of tables of IP address ranges which it can reach, either directly, or through another router. Users can specify *include address ranges* and *exclude address ranges* for use in these tables, to avoid sending out routes to known areas in the same network.

Each router sends a broadcast message (at periodic intervals) which includes these tables. Other routers receive the broadcast message, and determine from them, whether there is a better route to a particular IP destination, than the route stored in their *own* tables. If there is, they update their own tables. In this way, devices throughout the network(s) can

determine the best possible route for sending a message from one node to another. Various safeguards are built into the protocol to prevent looping situations where two routers each think the *other* router has the best route to a particular destination.

Inclusion Addr: This is an IP address, which will be used with the **Inclusion Mask** (below) to define a range of IP addresses which this controller will 'advertise' that it can reach, and so will be included in RIP broadcasts throughout the network. Note that this range may be further restricted based on the optional definition of an **Exclusion Addr** and **Exclusion Mask**.

Inclusion Mask: A non-zero value in any of the **Inclusion Mask** fields indicates that the corresponding **Inclusion Addr** field is specifying a portion of the IP address which must be identically matched with every IP address on routes which this controller is 'advertising' in its RIP broadcasts. A zero value in any of the **Inclusion Mask** fields means that any integer from (0 to 255) is considered valid *for that corresponding portion* of the Inclusion address.

Exclusion Addr: This is an IP address, which will be used with the **Exclusion Mask** (below) to define a range of IP addresses on routes which this controller will not *advertise* in its RIP broadcasts, because they are already known to be reachable (that is, they are in the same network). Note that this range can be further modified based on the optional definitions of an **Inclusion Addr** and **Inclusion Mask** discussed above.

Exclusion Mask: A non-zero value in any of the **Exclusion Mask** fields indicates that the corresponding **Exclusion Addr** field is specifying a portion of the IP address which must be identically matched with every IP address which this controller is specifically excluding from its advertised routes. A zero value in any of the **Exclusion Mask** fields means any integer from (0 to 255) is considered valid *for that corresponding portion* of the destination Exclusion address.

Important:

If you do not make any entries in either the Inclusion Addr/Mask or Exclusion Addr/Mask, RIP will NOT function.

Also, only devices which have been configured for RIP will be able to make use of the routing tables provided in the RIP broadcast messages.

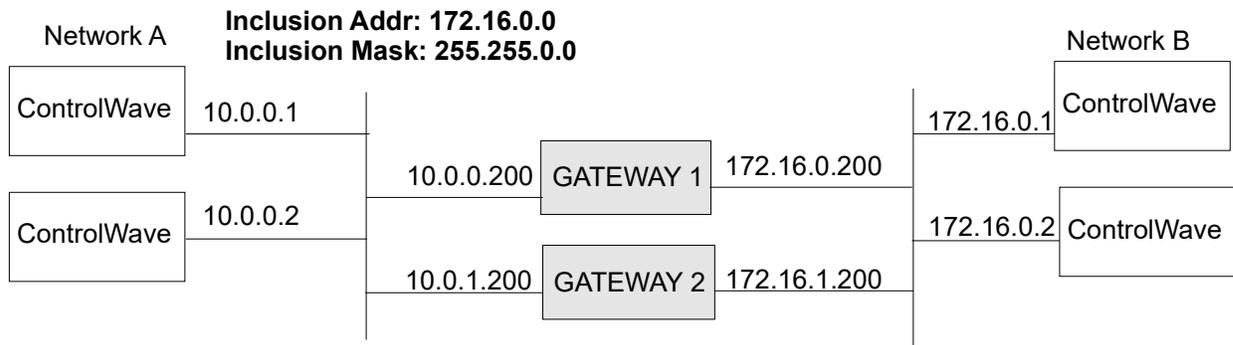
Some examples for setting the inclusion and or exclusion address/mask pairs are shown below:

In Example #1, on the next page, Network A, as well as Gateway 1 and Gateway 2 are all configured with RIP. Network B is not configured with RIP but has Gateway 1 as its default Gateway. Because of RIP, Network A will know about Gateway 2 as an alternate route to Network B, if Gateway 1 should fail.

Example #1 In this arrangement, Network A knows that both GATEWAY 1 and GATEWAY 2 provide a route to Network B. Should either GATEWAY fail, traffic to Network B can be routed via the other GATEWAY.

Network A and Gateways 1&2 support RIP.
Inclusion address/masks are set as follows:

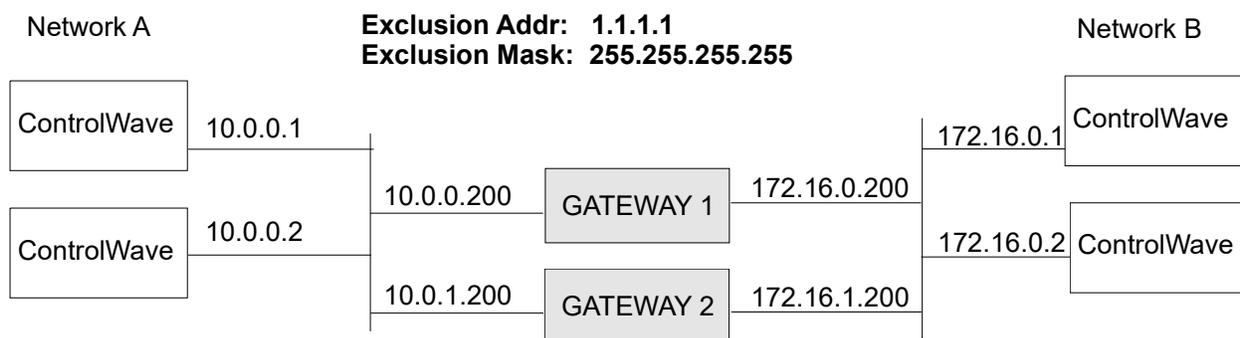
Network B does NOT support RIP, but has GATEWAY 1 as its default gateway.
Default Gateway:
172.16.0.200



In Example #2, Networks A and B, as well as the Gateways are all configured to support RIP. Here we specified just an Exclusion Address and Mask for an address which isn't even on any of the two networks. In this case we chose 1.1.1.1. With this minimal exclusion range defined, RIP broadcasts will include routes to ALL known addresses outside a particular Network, i.e. Network A will receive information about routes to Network B, and Network B will receive information about routes to Network A.

Example #2 - Network A, Network B, and all the gateways are configured for RIP, so that all routes between the networks are known.

Configure Exclusion Addr/Mask as follows:



Dynamic IP Routing Ping:

Dynamic IP routing is discussed in the description of the **IP Routes** page.

Rate: This is the frequency (in milliseconds) at which an IP route will be tested (via a ping message) to verify that the connection still functions. If the test is unsuccessful (no return from the ping within the specified timeout) the test is said to have failed. If **Retries** is a non-zero value, that number of additional attempts will be made to perform the ping test. If the test is still unsuccessful, IP traffic will be re-routed according to the information defined on the **IP Routes** page.

Timeout: This is period of time (in milliseconds) after which a ping test of a given IP route is said to have failed.

Retries: This is the number of additional attempts to perform a ping test will be performed after the first failure. If the total number of retries has been exhausted, re-routing of IP traffic will begin.

Challenge Protocol:

Two standard protocols have been implemented for security on PPP links in networks of ControlWave controllers: Challenge Handshaking Authentication Protocol (CHAP) and Password Authentication Protocol (PAP). These protocols operate in a client/server arrangement. Typically, CHAP should be used since it is more secure.

The CHAP (or PAP) server would be a ControlWave-series controller. The CHAP (or PAP) client could be either a ControlWave-series controller or an OpenBSI workstation.

The client must always supply a valid username/password combination in order to gain access to the server. If a ControlWave controller is the client, the username and password combination must have been pre-configured in the unit as a parameter stored in FLASH. This username / password text string will automatically be transmitted in response to a login prompt from the server.

Default Username: This is the username which will be transmitted if this ControlWave is serving as a PAP/CHAP client, and receives a challenge message from the PAP/CHAP server. This username must be one of the user accounts defined for the ControlWave, and will be sent along with the password defined for the specified user account.

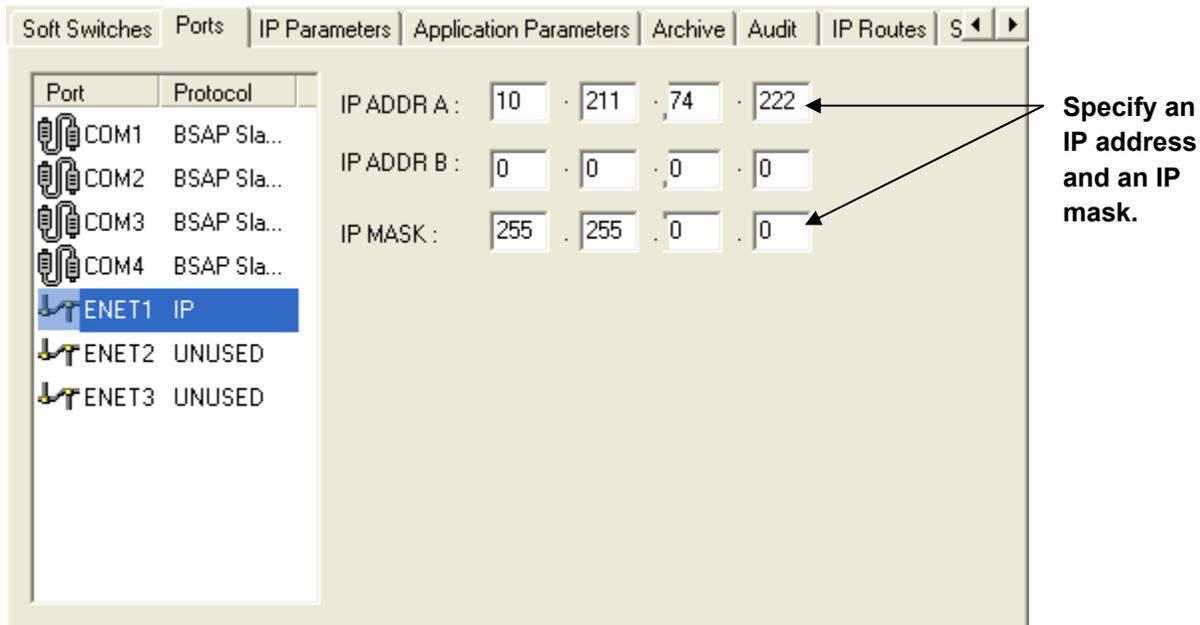
Click **[Write to RTU]** to save changes to the IP parameters. NOTE: Changes will NOT take effect until *after* the controller has been powered off and then back on.

IP Ports - Ethernet

A ControlWave-series controller's IP addresses are tied to its communication ports. There are two types of IP ports supported in ControlWave – Ethernet, and Point-to-Point Protocol (PPP). This section covers the Ethernet type:

Setting Up an Ethernet Port

Any of the Ethernet ports may be configured for IP communication. This configuration is performed from the 'Ports' page of the Flash Configuration Utility.



1. Choose the Ethernet port you want to configure.

Note:

If you will be defining more than one IP port (whether PPP or Ethernet) for this controller, it is strongly recommended that each IP port reside on a *separate* IP network. If you define more than one IP port on the *same* network, only one of the ports will be able to **send** messages; the other port(s) will only be able to **receive** messages.

2. Specify an IP address in the **IP ADDR A** field and enter an **IP MASK** for this port. IP addresses must be *unique* within your network. Conversely, IP masks are typically the same for all devices in the same portion of a network. Together, the IP address and IP mask define a range of addresses to which this port can send messages. (See *Recommended Ranges for IP Addresses* in the *IP Networks and Addressing* section.)

Note:

The **IP ADDR B** field only applies when configuring a redundant controller.

Basically, a non-zero value in any of the **"IP MASK"** fields indicates that the corresponding **"IP ADDR A"** field is specifying a portion of the IP address which must be identically matched with every destination IP address to which this port will send messages. A zero value in any of the **"IP MASK"** fields means that this communication port can send messages to addresses in which any integer from (0 to 255) is considered valid *for that corresponding portion* of the destination IP address.

Important

In newer ControlWave units, all Ethernet ports are pre-programmed at the factory with initial IP addresses and masks, as follows:

ETH1 IP Address: 10.0.1.1 IP Mask: 255.255.255.0

ETH2 IP Address: 10.0.2.1 IP Mask: 255.255.255.0

ETH3 IP Address: 10.0.3.1 IP Mask: 255.255.255.0

Because each unit shipping from the factory will have these initially pre-programmed, you should only use these addresses for 'bench' testing and configuration. *These addresses must be changed before putting ControlWave units on an actual network, since an address conflict would exist as soon as the second ControlWave unit was placed online.*

In the figure on the previous page, the **"IP ADDR A"** for the port is 10.211.74.222 and the **"IP MASK"** is 255.255.0.0. This means that this port can send to any address in the format 10.211.x.y where x and y are any integer from 0 to 255. So, 10.211.1.7 and 10.211.35.93 would be valid destinations, but 10.45.1.1, and 10.83.27.1 would NOT be because the 255.255 in the **"IP MASK"** indicates that the corresponding portion of the destination's IP address MUST be 10.211.

Important

If you accidentally specify overlapping address ranges for two or more Ethernet ports, ONLY the last Ethernet port created will function.

3. Click **[Write to RTU]**.
4. You must reset the controller to activate the new port configuration.

IP Ports – PPP

A ControlWave-series controller's IP addresses are tied to its communication ports. There are two types of IP ports supported in ControlWave: Ethernet and Point-to-Point Protocol (PPP). This section covers the PPP type.

Setting Up A Serial IP Port (PPP)

When the default switch is OFF, serial port COM1 on the ControlWave has an IP address of 1.1.1.1, and is configured for the serial point-to-point protocol (PPP). In any other configuration, PPP must be configured by the user.

Any of the serial COM ports can be configured as Serial IP (PPP) ports.

This configuration is performed from the **Ports** page of the Flash Configuration Utility.

Specify the baud rate for the port; this must match whatever baud rate you specify for the PC port.

Choose "PPP"

Port	Protocol
COM1	PPP
COM2	UNUSED
COM3	UNUSED
COM4	UNUSED
ENET1	IP
ENET2	UNUSED
ENET3	UNUSED

Physical Line Information

Baud Rate: 115200 Bits Per Char: 8

Stop Bits: 1 Parity: NONE

Protocol

Mode: PPP User Mode: 256

PPP

IP ADDR: 10 . 1 . 1 . 1

IP MASK: 255 . 0 . 0 . 0

Server Security

Plain Text (PAP)

Encrypted (CHAP)

None

Client Security

Plain Text (PAP)

Encrypted (CHAP)

None

Specify an IP address for this port

Specify an IP mask for this port

1. Choose the ControlWave port you want to configure.

Note

If you will be defining more than one IP port (whether PPP or Ethernet) for this controller, it is strongly recommended that each IP port reside on a *separate* IP network. If you define more than one IP port on the *same* network, only one of the ports will be able to **send** messages; the other port(s) will only be able to **receive** messages.

2. Choose 'PPP' from the "**Mode**" list box.
3. Choose the desired baud rate from the **Baud Rate** field. This must match the baud rate configured for whatever software you are using at the PC.
4. Either PAP or CHAP security can be configured on PPP lines. See the *Security* section in this manual for more information.
5. Specify an IP address in the **IP ADDR** field, and specify an **IP MASK** for this port. IP addresses must be *unique* within your network. Conversely, IP masks are typically the same for all devices in the same portion of a network. Together, the IP Address and IP Mask define a range of addresses to which this port can send messages. (See *Recommended Ranges for IP Addresses* in this manual for more information.)

Basically, a *non-zero* value in any of the **IP MASK** fields indicates that the corresponding **IP ADDR** field is specifying a portion of the IP address which must be identically matched with every destination IP address to which this port will send messages. A *zero* value in any of the **IP MASK** fields means that this communication port can send messages to addresses in which any integer from (0 to 255) is considered valid *for that corresponding portion* of the destination IP address.

In the figure on the previous page, the **IP ADDR** for the port is 10.1.1.1 and the **IP MASK** is 255.0.0.0. This means that this port can send to any address in the format 10.x.y.z where x, y, and z, are any integer from 0 to 255. So, 10.43.127.76 and 10.84.35.93 would be valid destinations, but 5.1.1.1 would not because the 255 in the **IP MASK** indicates that the corresponding portion of the **IP ADDR** MUST be 10.

6. If this PPP port uses PAP or CHAP protocol, choose whether the port is a PAP/CHAP server or a PAP/CHAP client, and select the appropriate protocol. See *Security Protocols* for an explanation of PAP and CHAP.. If the PC workstation is a server, you must configure the port as a client.
7. Click [**Write to RTU**].
8. Reset the controller to activate the new port configuration.

IP Routes

The IP Routes page is accessible from the **"IP Routes"** tab in the Flash Configuration Utility.

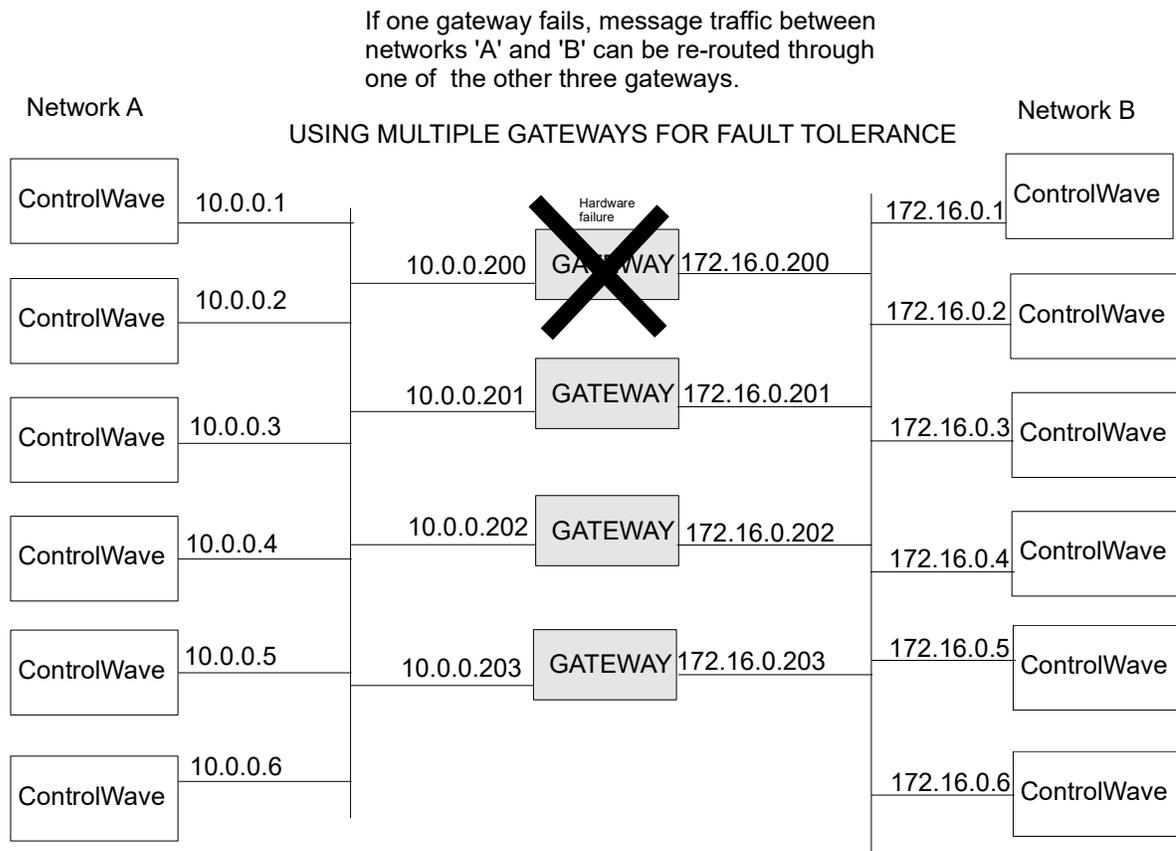
Beginning with ControlWave firmware CWP02.0, multiple gateways (routers) can optionally be configured for a particular network to support dynamic IP routing.

A **dynamic IP route** is considered to be a range of destinations (IP addresses) and the gateways used to reach them.

Gateways are essentially routers (devices which have IP connections on two or more separate networks). As such, they provide a means for sending messages from one network to another. You might want to think of gateways as entrance ramps to a highway.

Up to 4 gateways can be configured to reach a particular destination address range, and each controller can have up to 16 destination address ranges specified.

Since messages can be sent to a particular route by a choice of more than one gateway, the system can attempt transmission through one gateway, and if it fails, traffic will be sent through one of the *other* gateways. This provides a degree of fault-tolerance in the system (see the following figure).



The system can test a particular path by using a specified ping address. The ping address could be the gateway itself, or it could be the destination controller.

The actual re-routing occurs only after a specified timeout has expired. (See the *IP Parameters* section in this manual for details.)

The 'IP Routes' page displayed, below, shows a typical configuration for the network depicted on the previous page. (This configuration would be for a controller belonging to network "A" as shown on the previous page.)

The screenshot shows the 'IP Routes' configuration page. At the top, there are tabs for 'Soft Switches', 'Ports', 'IP Parameters', 'Application Parameters', 'Archive', 'Audit', and 'IP Routes'. Below the tabs is a table with two columns: 'Route' and 'Destination'. The first row is selected, showing '1' in the 'Route' column and '172.16.0.1' in the 'Destination' column. To the right of the table are configuration fields for 'Route 1 Destination', 'Route 1 Gateways', and 'Route 1 Pings'. The 'Route 1 Destination' section has 'IP Address' set to 172.16.0.1 and 'IP Mask' set to 255.255.0.0, with a checked 'Check Primary' box. The 'Route 1 Gateways' section has four 'IP Address' fields set to 10.0.0.200, 10.0.0.201, 10.0.0.202, and 10.0.0.203. The 'Route 1 Pings' section has four 'IP Address' fields set to 172.16.0.200, 172.16.0.201, 172.16.0.202, and 172.16.0.203.

After you have completely defined a particular route, you can click on the next route number in the box in the upper left corner, and the various fields will be cleared to allow you to enter information on the next route. A total of 16 separate routes can be defined.

Route x Destination

- IP Address:** This **IP Address** together with its **IP Mask** define a range of destination IP addresses on this particular route.
- IP Mask:** Any non-zero value in the **IP Mask** specifies a portion of the IP address which must be identically matched with every IP address on the destination route.
- Check Primary:** In the event re-routing has occurred due to a failure, checking this selection will force a re-test of the first gateway (or ping address) to see if the failure has been corrected, thereby allowing traffic to return to its normal path via the first gateway. This might be particularly important if the secondary route is *slower* than the primary. If this is NOT checked, traffic will continue to use the secondary route, unless it too fails.

Route x Gateways:

IP Address 1 to IP Address 4: These must be IP addresses for gateways which are in the same network as the current controller. During normal operation, the gateway 1 address would be used, but if there is a failure along the path defined for that gateway, an attempt will be made to re-route traffic to the *next* gateway (IP address 2). If that second gateway cannot be used, then the next would be used, and so on, up to the fourth gateway. If the last configured gateway fails, an attempt will be made to use the first gateway, and so on.

Route x Pings:

IP Address 1 to IP Address 4: For each of the four possible paths of a given route, the user can optionally define a ping address for testing the route. Typically, the ping address would be the IP address used by the gateway connection in the *other network*. Alternatively, the ping address could be one of the destination controllers in the other network; this might be done to check for failure of one of the controllers in a redundant pair.

Libraries

A **library** is a collection of functions and function blocks, which can be used in a ControlWave Designer project. There are two types of libraries: firmware libraries and user-created libraries.

Firmware libraries include functions and function blocks which are defined in the internal system firmware of the ControlWave controller. They have the file extension, *.FWL, and are loaded automatically when you create a project using the ControlWave template. Two examples of firmware libraries are the **ACCOL3** library, which includes all of the ACCOL3 functions and function blocks, and the PROCONOS library, which includes various KW standard function blocks, and IEC 61131 standard function blocks. NOTE: For information on the various functions and function blocks in the ACCOL3 and PROCONOS libraries, please consult the online help in ControlWave Designer.

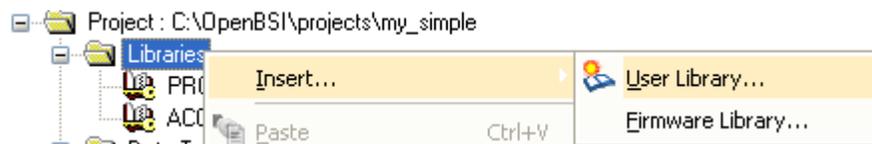
User libraries include functions and function blocks, which you have created yourself, for some application-specific purpose. User created function blocks are made by combining, in some logical way, functions and function blocks that already exist, typically from the ACCOL3 or PROCONOS firmware libraries. User libraries have a file extension of *.MWT, and can be re-used in any ControlWave project, by inserting the library in that project.

Creating a Library of User-defined Function Blocks

Note

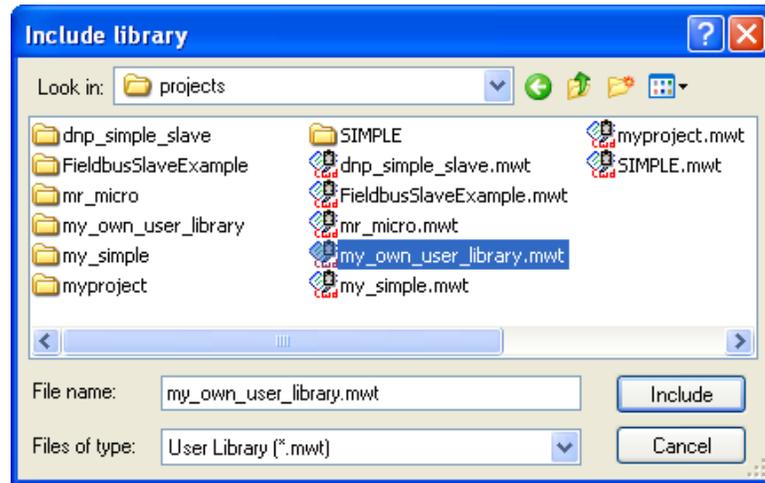
For information on creating user-defined function blocks (which you must do **before** you can create a user library) see *Function Blocks - Creating* in this manual.

1. Choose an existing project (*.MWT file) which includes the user-defined function blocks you want as part of your user library. If desired, rename that project to reflect the name you want for your library. In this case, we have named our library 'My_own_library'.



2. Open whichever project you want to include the library in. Right click on the "Libraries" icon and choose "Insert→User Library" from the pop-up menus.

3. Select the MWT file from step 1. This is called *announcing the library*.



4. The next time you call up the Edit Wizard, the library should appear as a group, from which you can select your user-defined function blocks.



Manually Including the ACCOL3 Firmware Library in Your Project

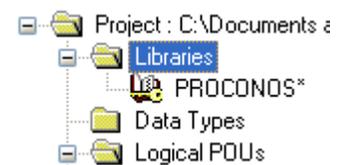
Note

If you are unsure whether the ACCOL3 Firmware Library is part of your project, look in the “Libraries” folder in your project tree. If you see ‘Accol3’ in the tree, you already have the ACCOL3 Firmware Library included. You can then skip this section.

Note

If, when you first opened your new project, you chose the ControlWave template, the ACCOL3 firmware library (which includes the ACCOL3 function blocks) was automatically included in your project. You can then skip this section.

If you did *not* choose the ‘ControlWave’ template when you opened your new project, and you want to use ACCOL3 function blocks as part of your project, you **must** include the ACCOL3 Firmware Library.

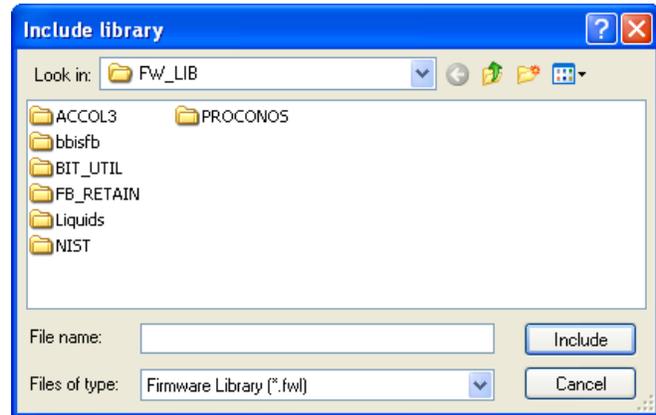


Note

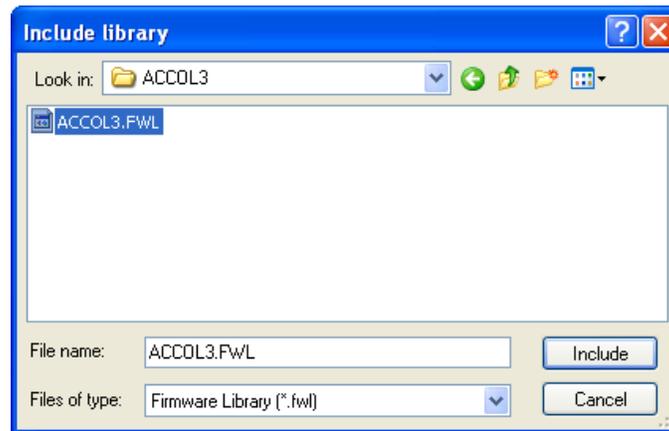
All parameters for the function blocks in the ACCOL3 Firmware Library are forced by the system to be RETAIN variables.

To do this, *right click* on **“Libraries”** in the project tree, and choose **“Insert”** from the pop-up menu. The Include Library dialog box will appear.

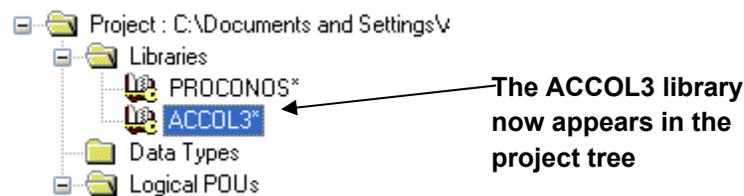
Change the **“Files of type”** list box to show ‘Firmware Library (*.fwl)’.



Go to the folder, ‘\OpenBSI\Mwt\Plc\FW_lib\Accol3\’ and click on ‘Accol3.fwl’ in the list box, then click on the **[Include]** button.



The ACCOL3 Firmware library will now appear in the project tree, under **“Libraries”**.



Memory Usage

This section explains the terminology used in the discussion of ControlWave memory, and explains the differences between the different kinds of memory used in the ControlWave series of controllers.

Note

This section describes the memory for a standard ControlWave first. The internal memory arrangement and usage for other ControlWave products (Micro, XFC, EFM, GFC, Express, CW_10, CW_30, CW_35) are *different*. Variations between the different products are discussed near the end of this section.

Some Background - What is Memory?

As a ControlWave controller runs its program(s), the controller's central processing unit (CPU) executes each instruction in the program. These instructions, and the data associated with them, are read from, and/or written to, physical locations within computer chips in the controller. These physical locations are referred to as **memory**. They are similar to memory you might have in your personal computer at home. Each memory location also has a numerical identifier called an **address**. The address is used internally to locate data stored in memory.

The amount of memory in your controller varies depending upon the purchased memory options.

Information in memory is stored as a series of 0s and 1s; each '0' or '1' is referred to as a **bit**. These bits are grouped together into chunks of 8 which are called **bytes**. (Two bytes together are called a *word*.) Each byte can hold a character of data. A group of 1024 bytes is referred to as 1K. 1024K is referred to as a Megabyte or 1MB.

What is Downloading?

Downloading is the process of transferring the compiled control strategy from your PC workstation, into the memory of the ControlWave controller. Additionally, you can also download the compressed project source code (*.ZWT).

In ControlWave Designer, you create a control strategy that directs the controller to perform a system-specific job and save that strategy as a ControlWave **project**. The project is compiled and the resulting computer code is then **downloaded** from the PC into the ControlWave's memory using either ControlWave Designer or the OpenBSI Downloader.

When downloading the project directly from within ControlWave Designer, you have a choice of downloading the project directly into the dynamic memory, or downloading into the *flash* memory area (this is called downloading the **bootproject**). When downloading from the OpenBSI 1131 Downloader, only the bootproject may be downloaded.

A project can only *execute* from the dynamic memory area, but is lost in the event of a power failure or a program watchdog condition.

Because dynamic memory is not saved in these situations, a copy of the project is typically stored in the bootproject area of flash memory. The bootproject cannot execute from inside the flash memory area, but is automatically copied into the dynamic memory area during system re-boot (either from a power restoration after power failure or a restart following program watchdog condition).

Typically, you download a project into the dynamic memory area only during system development and debugging. Once a control strategy is finalized, and has been tested fully, it should be downloaded as the bootproject into flash memory.

Instructions for downloading from within ControlWave Designer are included in the section *Downloading with ControlWave Designer* in this manual. Instructions for downloading using the OpenBSI Downloader are included in Chapter 7 of the *OpenBSI Utilities Manual* (part number D301414X012).

Types of Memory in the ControlWave Process Automation Controller (CW PAC)

- The system supports four types of memory which are of interest to the user:
- Boot flash
- Synchronous Dynamic Random-Access Memory (SDRAM)
- Static Random-Access Memory (SRAM)
- Flash

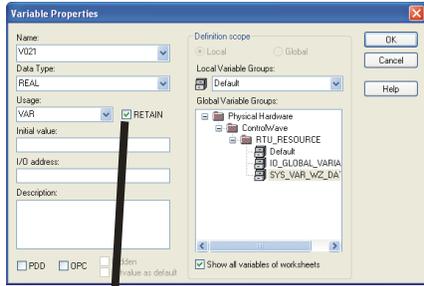
Note

In addition to the memory types listed here, the CPU contains its own CMOS RAM for the real-time clock and configuration data. None of this memory can be directly used by your ControlWave program(s).

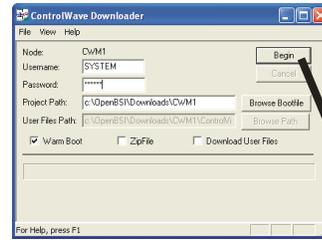
The following table details the types of information stored in these types of memory.

Type of Memory	Amount Currently Supported	Usage
Boot Flash	512K	<ul style="list-style-type: none">▪ Soft switch values.▪ User account and port configuration parameters.▪ Archive File Definitions▪ Audit Trail Configuration Parameters

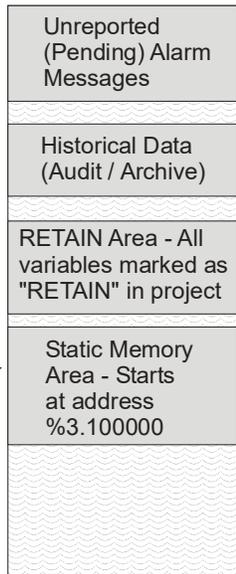
Type of Memory	Amount Currently Supported	Usage
SDRAM	64 MB	<ul style="list-style-type: none"> ▪ The running control strategy (ControlWave project) and the current values of any variables NOT marked as RETAIN, or stored in the Static Memory area. ▪ A copy of the system firmware (loaded from FLASH into SDRAM to allow for faster system execution). <p>Notes:</p> <ul style="list-style-type: none"> ▪ The project and system firmware can ONLY execute in SDRAM. ▪ Anything stored <i>only</i> in SDRAM is lost in the event of a watchdog failure or power failure.
SRAM	2 MB	<ul style="list-style-type: none"> ▪ Current status of output variables in RETAIN. ▪ Historical data (Audit Trail / Archive) (Optional - most users choose to store this information in FLASH instead) ▪ Current values for any variables marked as 'RETAIN' including the variables associated with ACCOLIII function blocks. ▪ Values for variables manually assigned to the Static Memory Area (beginning with addresses %3.100000). The static memory area is only initialized at a system cold start (discussed later). ▪ Pending alarm messages (alarms which have not yet been reported to the user) <p>Notes:</p> <ul style="list-style-type: none"> ▪ So long as switch SW1-5 is set ON, and the backup battery continues to operate, data stored in SRAM is retained in the event of a system warm start. This memory is initialized, however, if switch SW1-5 is set OFF, or if the backup battery fails.
Flash Memory	32 MB	<p>The ControlWave system firmware (firmware uses up to 4MB). The remaining memory can hold the following:</p> <ul style="list-style-type: none"> ▪ The ControlWave bootproject ▪ The compressed ControlWave source code file including graphical elements of the project (*.ZWT) ▪ Historical data (Audit Trail / Archive) <p>Notes:</p> <ul style="list-style-type: none"> ▪ The flash memory area is unaffected by a power failure or watchdog condition.



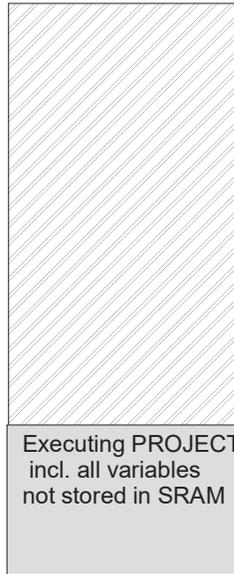
- BOOT FLASH
512 K
- Audit/Archive Def
- Soft Switches
- User Accounts
- Port Config.



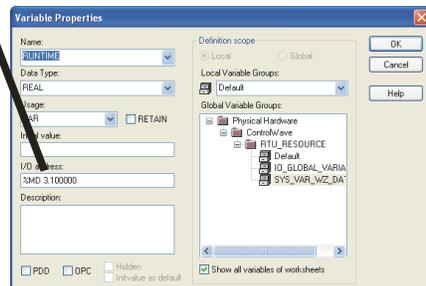
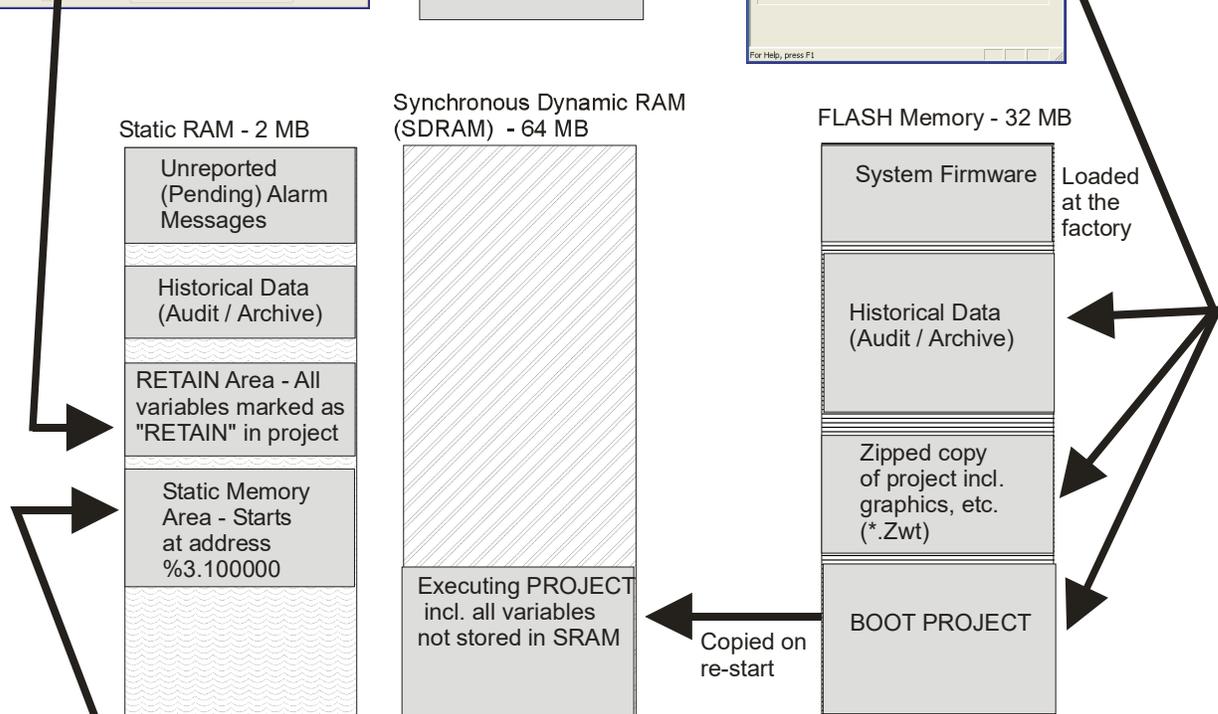
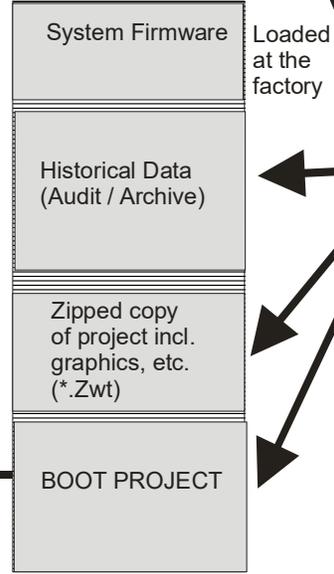
Static RAM - 2 MB



Synchronous Dynamic RAM (SDRAM) - 64 MB



FLASH Memory - 32 MB



What happens in the event of a power failure or the power switch is turned off?

If the ControlWave loses power, or the power switch is turned off:

- The synchronous dynamic RAM (SDRAM) area is cleared. The executing control strategy, and its running data are lost.
- If the SRAM Control Switch (SW1-5) is set to ON *and* the backup battery for the SRAM remains good, data is preserved. If you have made edits to your project that directly affect variables marked 'RETAIN', the RETAIN area may be erased when the system restarts, because the structure of the RETAIN area has been modified. Other areas of SRAM (such as the Static Memory Area or historical data) **are not** erased.
- Flash and BOOT FLASH memory will be preserved.

The controller will attempt to restart immediately whenever power is restored. (See *What happens on restart after a power failure or watchdog?*).

What happens in the event of a watchdog condition?

If the ControlWave suffers an internal program error (but power remains good) which causes the executing control strategy to halt (called a *watchdog condition*):

- The synchronous dynamic RAM (SDRAM) area is cleared. The executing control strategy, and its running data are lost.
- Any analog I/O boards which are configured with hold values will hold their last value at the output pins, provided that the I/O board still has power. NOTE: Only certain analog I/O boards support this feature.
- If the SRAM Control Switch (SW1-5) is set to ON, *and* the backup battery for the Static RAM (SRAM) remains good, data is preserved. If you have made edits to your project that directly affect variables marked 'RETAIN', the RETAIN area may be erased when the system restarts, because the structure of the RETAIN area has been modified. Other areas of SRAM, i.e. the Static Memory Area, historical data, etc. will not be erased.
- Flash and Boot flash memory is preserved.
- The Watchdog hardware relay/switch circuitry will be activated (this can be wired to an external device (klaxon, alarm bell, etc.) to indicate that the controller has entered a watchdog condition.

The controller will attempt to restart immediately after entering the watchdog state. (See *What happens on restart after a power failure or watchdog?*).

What happens on restart after a power failure or watchdog?

When a ControlWave controller first starts up, it first performs various diagnostic checks ("is the power stable?" etc.). The results of these checks are displayed on the power-on self test (POST) display. Internal system devices are then initialized.

Under normal operation (no system firmware upgrade needed) the system then decides whether it can perform a system warm start or a system cold start.

Note:

System firmware is loaded into the ControlWave before it leaves the factory. If you need to perform a field upgrade of system firmware (new version released with new features, or update needed to correct some problem), set switch SW3-3 **on before** powering on the ControlWave. The ControlWave then enters recovery mode and shows "86" on the display. You can then load system firmware (for directions, refer to the *ControlWave Process Automation Controller Instruction Manual*, part number D301381X012).

System warm start or system cold start

A **system cold start** occurs when the SRAM control switch (SW1-5) is set to OFF or the backup battery for the SRAM has failed. In either of these cases, the entire static RAM (SRAM) area is erased. Any other situation is referred to as a **system warm start**.

Load System Firmware, Start Communications

Once the system warm start or system cold start completes, the system firmware is copied from flash memory into SDRAM, because firmware can only execute from SDRAM.

Next, the communication system starts. Soft switches are read, and the communication ports are activated.

Next, the ControlWave checks to see if there is a project in the bootproject area of flash memory. If no project is present, the ControlWave shows "00" on its display and waits for you to download a project. If a project is present, the bootproject will be copied into dynamic memory SDRAM to allow it to be executed. The system then determines whether to perform an application warm start or an application cold start.

Application Warm Start or Application Cold Start

An **application warm start** means that the project in SDRAM starts from the beginning of its cycle, using saved values for variables marked "RETAIN" from the static RAM (SRAM). Application warm starts are performed whenever there is no version mismatch between the project, and the retain values, and there was no **system cold** start.

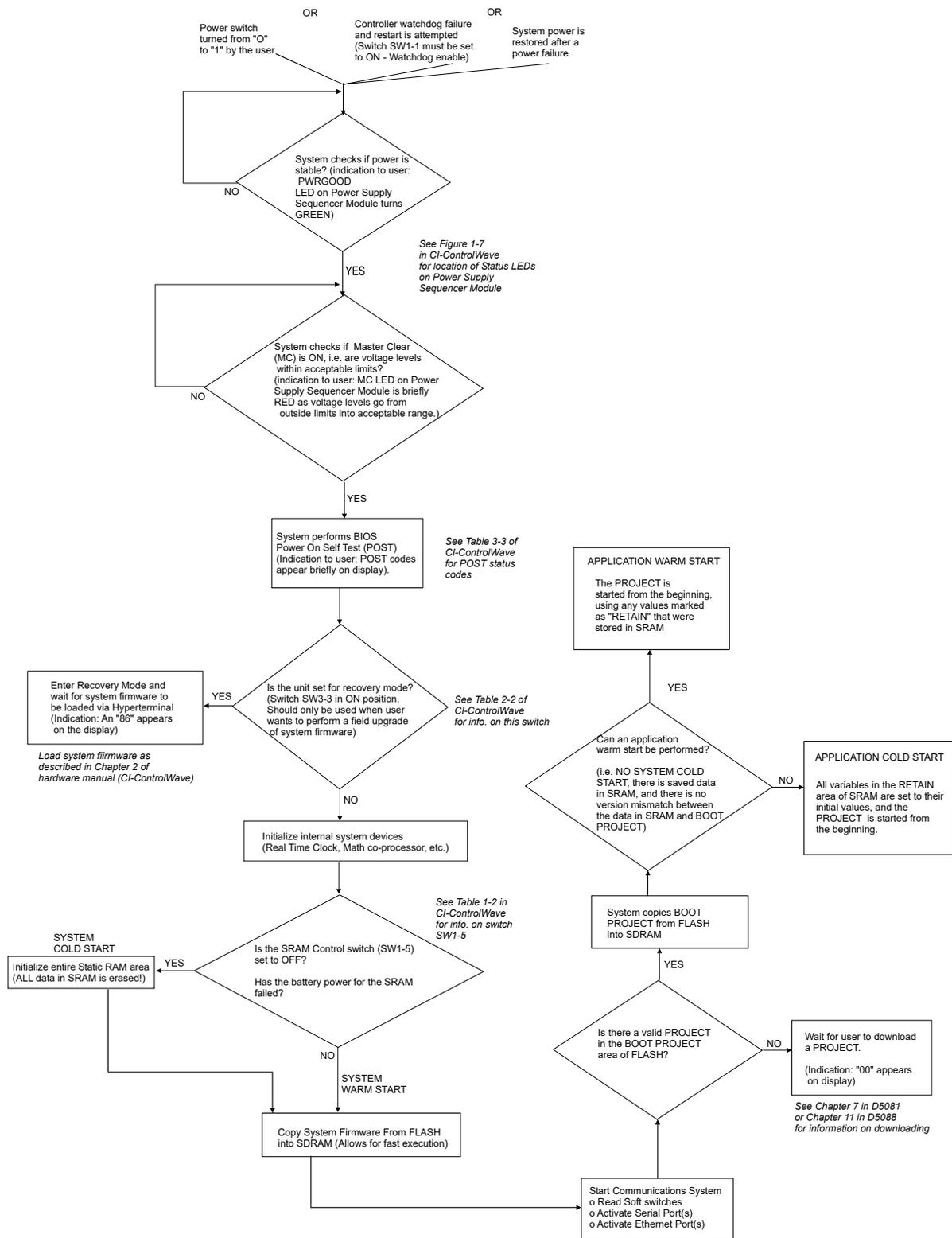
If a **system cold** start occurred (because of loss of battery power to the SRAM or the switch SW1-5 was set to off), all data in static RAM is gone, so an **application cold start** must be performed.

In this case, the project in SDRAM is started from the beginning, and all variables are set to their initial values.

You can also perform application warm starts and cold starts on demand after downloading a project from within ControlWave Designer by clicking the **[Cold]** or **[Warm]** buttons in the RTU Resource dialog box.

The figure on the next page shows the start-up sequence of the ControlWave controller following a watchdog or restoration of power.

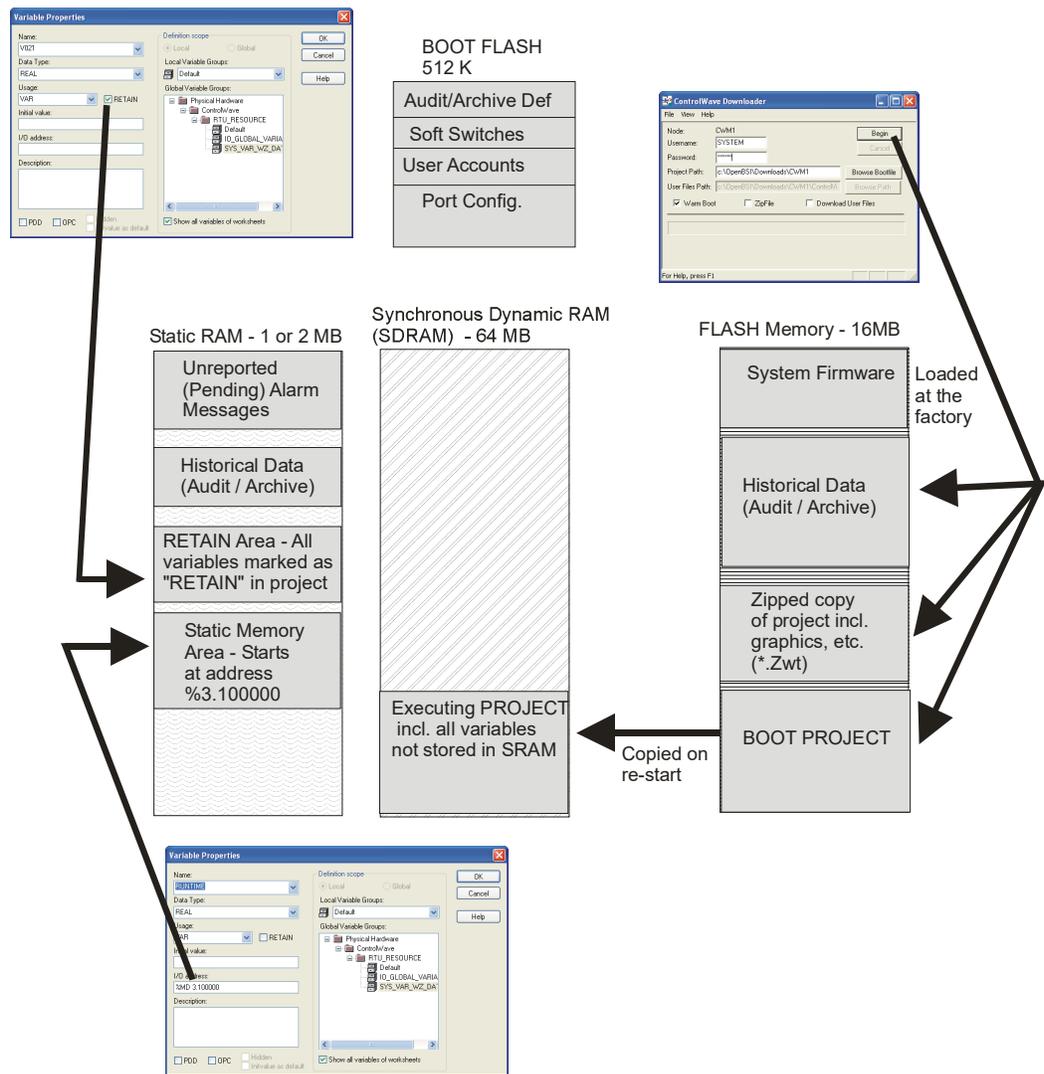




Variations when using ControlWave MICRO/EFM

The main difference between the memory configuration in the ControlWave and the ControlWave MICRO and EFM is that the flash memory in the MICRO is faster, so system firmware that would have been executed in SDRAM executes in flash memory instead.

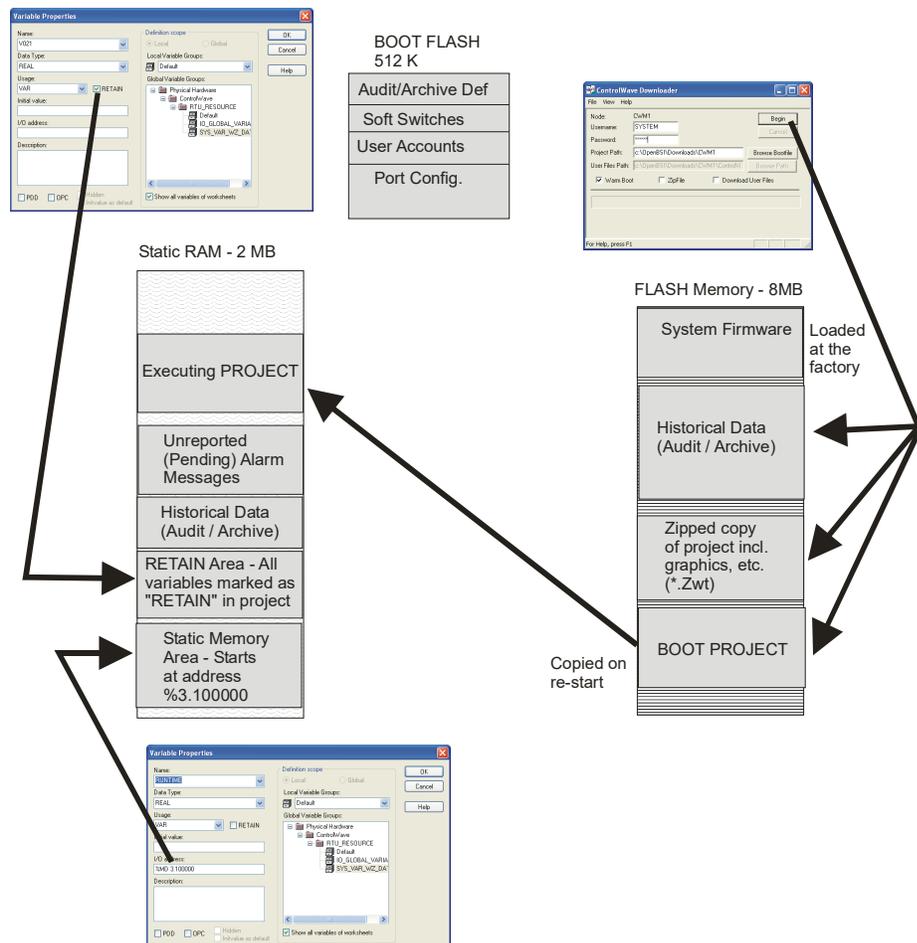
- Executing system firmware resides in flash memory; it is NOT copied into SDRAM. (The control strategy itself *does* execute in SDRAM.)
- The SRAM Control Switch is SW2-5.
- The Recovery Mode Switch is SW1-3.
- The Watchdog Circuit Enable Switch is SW2-1.
- The ControlWave Micro and ControlWave EFM have 16MB flash instead of 32MB in the ControlWave PAC.



Variations when using ControlWave GFC/GFC-CL, XFC, Corrector, Express or ExpressPAC

The main difference between the memory configuration in the ControlWave and these units is that the flash memory in these units is faster, so system firmware executes in flash memory. Also, the project executes in SRAM, because **there is NO SDRAM in these units**.

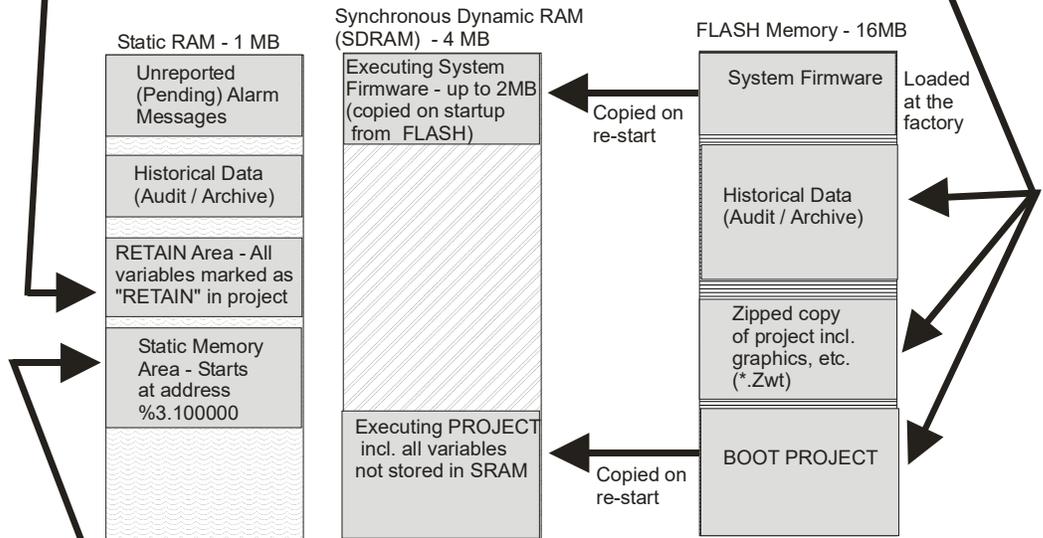
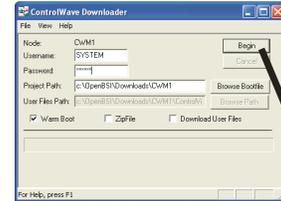
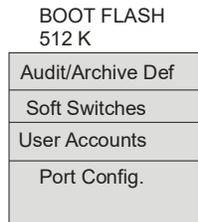
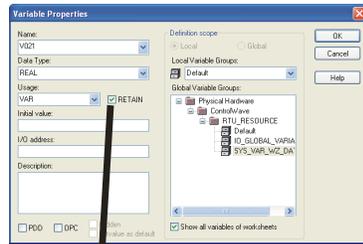
- Executing system firmware resides in FLASH memory.
- The Control strategy itself executes in Static RAM (SRAM). On restart, the control strategy would be lost.
- The SRAM Control Switch for these units is SW2-5, except for the XFC which uses SW1-5.
- The Recovery Mode Switch is SW1-3 for these units except for the XFC which uses; SW1-9 and 10.
- The Watchdog Circuit Enable Switch for these units is SW2-1, except for the XFC which uses SW1-1.



Variations when using ControlWave_10/ _30/ _35 (CW_10, CW_30, CW_35)

Memory in the CW_10, CW_30, and CW_35 is very similar to the ControlWave Process Automation Controller. There are some differences, however:

- There is less static RAM (1 MB of SRAM instead of 2MB in the ControlWave).
- There is less flash memory (16 MB instead of a maximum of 32MB in the ControlWave).
- The SRAM Control Switch is SW2-5.
- The Recovery Mode Switch is SW1-3.
- The Watchdog Circuit Enable Switch is SW2-1.



Memory Allocation Issues

The ControlWave Process Automation Controller contains 64 MB of SDRAM. Of this 64 MB, approximately 700 KB is required to hold the system firmware (the internal code which tells the CPU how to run programs), and another 300 KB is required for system overhead.

When all other things are taken into account, approximately 62.7 MB of SDRAM are available for your ControlWave project, and its data. The only *fixed* restrictions on how you use the remaining 62.7 MB are:

- No program organization unit (POU) can exceed 640 KB. (Prior to OpenBSI 5.7 Service Pack 2, this limit was 64K.)
- There can be no more than 512 POUs in the project.
- A small amount of free-space (between 64 KB and 128 KB must be maintained).

Determining POU Size at Compilation Time

As previously noted, any POU exceeding 640 K bytes **will not** work. When you compile your project using the **Make** command, you can determine the *approximate* size of individual POUs by calling up the **"Infos"** tab in the Message Window and multiplying the number of bytes shown for a POU by a platform-dependent factor. For standard ControlWave units (using the IPC processor) use a factor of **1.3**. For other units (using the ARM processor) use a factor of **1.5**. Note that this is just an approximation, the actual size may vary. Any POU exceeding the 640 Kb size limit will either need to be removed or re-written so that it does not exceed the 640 Kb limit.

Resolving “Not Enough Memory” Messages

If you encounter error messages related to not having enough memory after downloading the project into the ControlWave controller (or into the I/O Simulator), your project is too big as currently configured. There are a few things you can do to reduce the size of the project, so that it may fit into the available SDRAM space:

Find out how much SDRAM your project is using, so you know how much you need to get back

When you download a project into the ControlWave, or into the I/O Simulator, statistics are maintained as to how much memory is free, after the download. To view this information, click **[Info]** in the RTU_RESOURCE dialog box.

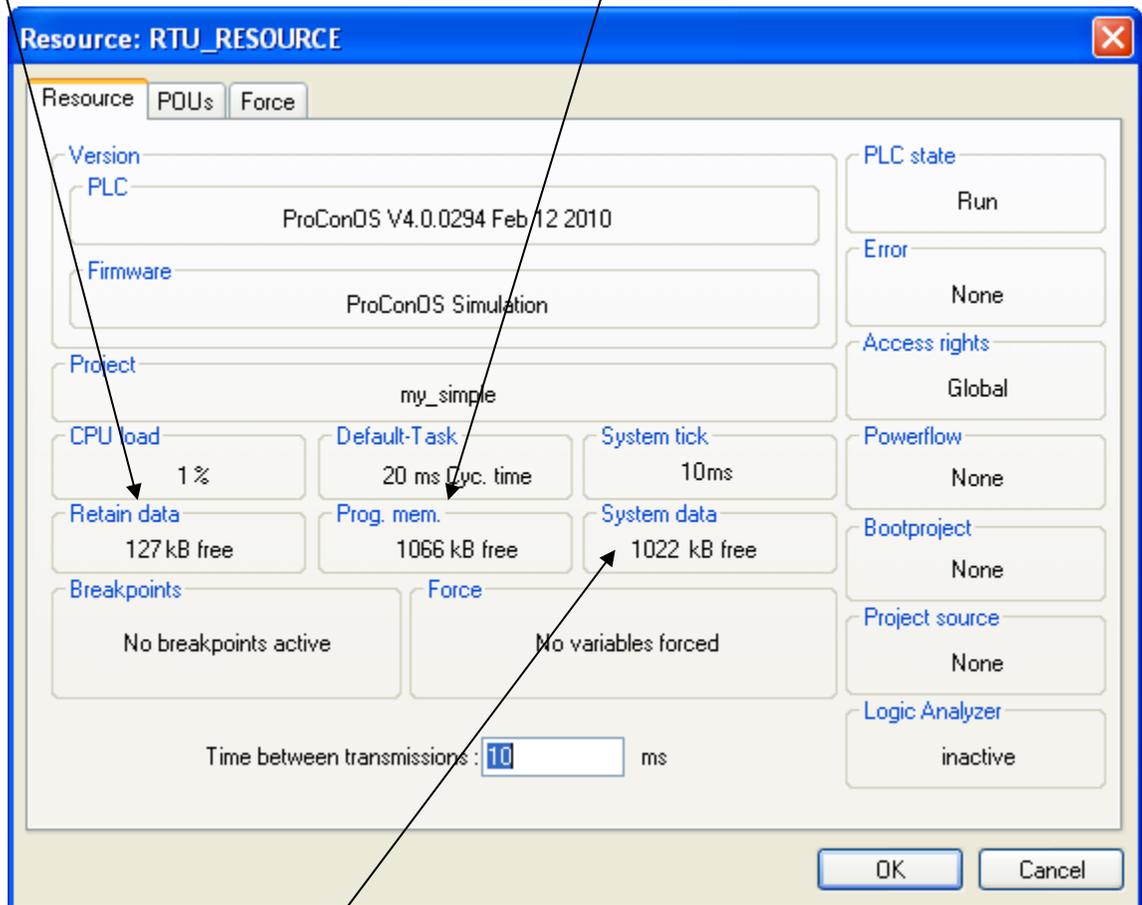
Click here to obtain statistics on memory usage.



The 'Resources' tab displays the amount of memory available:

This is the amount of RETAIN memory available in SRAM (static RAM).

This number indicates the amount of SDRAM (or SRAM if the unit does not support SDRAM) available to run program POU's.



This is the amount of SDRAM (or SRAM if the unit does not support SDRAM) available for anything other than your program POU's such as communication buffers, PDD, system firmware.

The POU's tab displays the amount of memory used, including the total program size, and the amount of data used for both RETAIN and non-RETAIN variables.

Number of bytes used by code for particular functions, function blocks

Name	Code	NonRetain: Data	Reserve	Retain: Data	Reserve
ALARM_ANALOG	146	82	0	62	0
CTD	273	10	0	0	0
CTU	270	10	0	0	0
CTUD	433	12	0	0	0
Flow_Control_Program	717	446	100	112	10
LEAD_LAG	144	14	0	12	0
PID3TERM	144	30	0	24	0
TOF	392	26	0	0	0
TON	370	26	0	0	0
TP	402	26	0	0	0
Global variables	-----	460	100	0	0
INIT_CODE(internal)	653 (1%)	-----	-----	-----	-----
TASK_MODULE(internal)	121	-----	-----	-----	-----
Total	4065	906	-----	112	-----

PDD Size
Total: 50
Relative: 0%

Export to CSV file ...
Copy to clipboard

OK Cancel

Total program size (in bytes). NOTE: May not be equal to the sum of individual lines due to system overhead.

Number of bytes of data (both RETAIN and non-RETAIN) used by the variables for these functions/function blocks.

Note:

Certain function blocks are always included in memory, even if you don't use them in your project. These include counter function blocks (CTD, CTU, CTUD) and timer function blocks (TOF, TON, TP).

If possible, reduce the number of variables marked “PDD”

Checking the “PDD” box for a particular variable stores the variable’s name in memory, which allows allow external programs (such as OpenBSI’s DataView) to collect that variable.

If you check “PDD” for every variable (just in case you might want to collect it at some later time), this forces all of those variable names to be stored in memory. If you have *thousands* of variables marked for PDD, this can result in a shortage of available memory.

Examine the variables that you have marked for “PDD” collection and see if there are some you could *de-select*.



Initialize LIST function blocks via DB_LOAD, if possible

If you have a large number of LIST function blocks, containing *thousands* of list elements, which you initialize within the project, you should consider performing the initialization using the DB_LOAD function block. DB_LOAD allows much of this information to be stored in a text file, external to the project, thus reducing the amount of memory used. For more information on DB_LOAD, see the ACCOL3 help files in ControlWave Designer.

Adjust Application Parameters for Memory

Application parameters for memory are changed from the ‘Application Parameters’ page in the Flash Configuration Utility.

Important

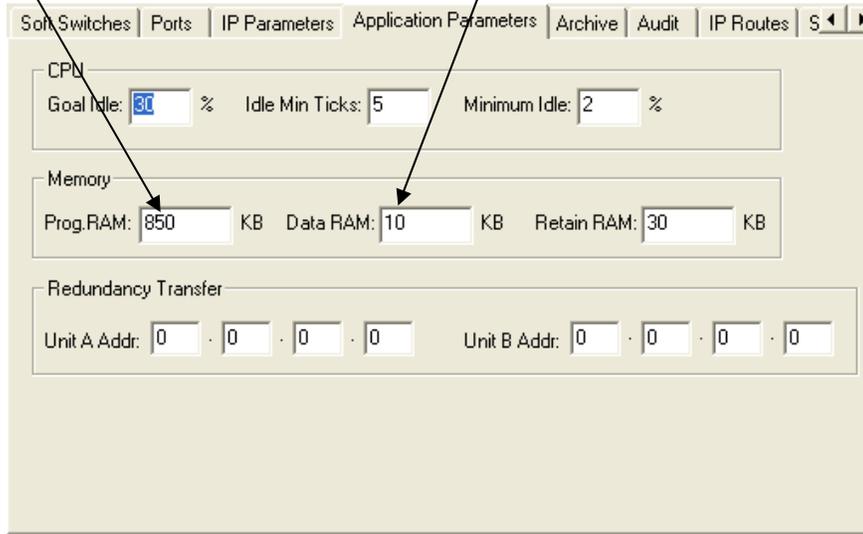
Users should exercise caution when modifying the application parameters for memory. Making a significant change to these parameters without understanding how the parameters interact could actually reduce the amount of available memory, even though you have increased the values of the parameters. For more detailed information about the individual application parameters, see the *Application Parameters* section earlier in this manual.

Before changing any of the application parameters for memory, you should first check how much memory you have free (from the ‘Info’ option of the RTU_RESOURCE dialog box.)

If you have little or no “Prog. Mem” free, you should increase slightly the “Prog RAM” value in the Application Parameters page of the Flash Configuration utility. Changes should be incremental; don’t set “Prog RAM” too high - its value is fixed on startup. If you set it significantly larger than it needs to be, all the extra space will be wasted because it will be reserved for the program, even if the program isn’t large enough to make use of the space. That unused space will be missing from the available memory pool for the data, which could result in memory errors. If, however, you set the value for “Data RAM” to be too large, the system will reduce it, proportionally, as needed.

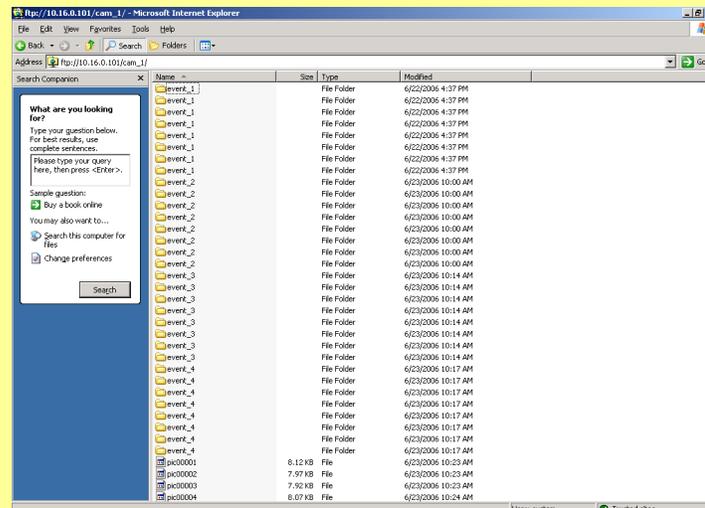
If you set this value to a number that is significantly larger than you need, you will waste the excess amount.

If you set this value too large, the system reduces it proportionally, as needed.



Notes about Flash Files and Folders

Flash memory in the ControlWave uses a linear file structure. Because of this, the folders are used internally just as path names for files. If you were to use a File Transfer Program (FTP) to examine the contents of the ControlWave's flash memory, you would see what appear to be multiple folders sharing the same name. Don't be concerned by this; they are not duplicate folders, just multiple references to the same folder. The picture, below, shows this duplication. Several folders share the same name; in reality these are simply unique references to the same folder *for each file* in the folder.



Modbus Configuration

Configuring Your ControlWave Controller as a Modbus Master Device

Important

This example assumes you are familiar with Modbus concepts.

Modbus is an industry-standard communications protocol used by a wide variety of controller and PLC manufacturers. By configuring your ControlWave unit as a Modbus Master device, you can read/write coil status, 16-bit register, or 32-bit register information from a PLC serving as a Modbus Slave, and store that data in variables or structures in your control strategy program.

Important

Modbus communication requires that you include the CUSTOM function block in an **executing** task. Modbus communication requests/responses and processing only occur when the CUSTOM function block executes. Also, you must have a port configured for Modbus communication, before you begin.

Note

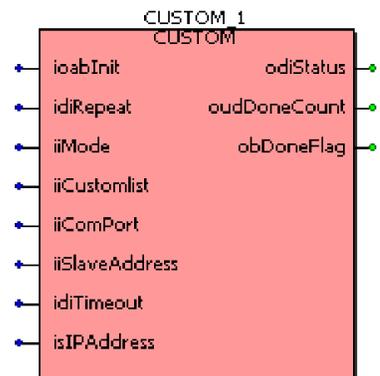
Both serial Modbus communication and Open Modbus (IP) communication is supported. Serial Modbus communication utilizes the serial ports, Open Modbus (IP) communication utilizes the IP ports. For help on configuring a communication port for Modbus, see the *Modbus Ports* section.

Step 1. Insert the CUSTOM function block in your program.

The program you choose must have been assigned to an executable task. The system will attempt to perform Modbus communications at the rate specified for the task containing the CUSTOM function block.

Step 2. Configure the CUSTOM function block for Modbus Master communications.

The following table describes the various parameters of the CUSTOM function block, as it is used in this particular example:



Parameter Name	Type of Variable	Description
ioabInit	BOOL input/output variable (NO constants allowed)	Init - This parameter should be set to TRUE any time a configuration change has been made to any of the input parameters (except for changes in the IOList discussed later). This re-initializes the function block with the new configuration data. Once the re-initialization is complete, this parameter will automatically be set to FALSE. <i>NOTE: The very first time the function block is executed, this parameter is automatically set to TRUE in order for initial configuration parameters to be set.</i>
idiRepeat	DINT input	Repeat - This optional parameter specifies, in milliseconds, the time which must expire before another attempt will be made to send/receive a Modbus message. So, for example, if 0 is entered as the repeat value, an attempt will be made to send/receive a Modbus message every time the CUSTOM function block executes (provided that the previous message has completed). If 10000 is entered as a repeat value, the executing CUSTOM function block will attempt to send/receive a MODBUS message only if 10 seconds have elapsed since the last Modbus message request.
iiMode	INT input	Mode – Specifies the mode in which the Custom Block operates. Valid values are: 4 (Modbus Master – serial) or 52 (Open Modbus Master (IP)).
iiCustomList	INT input	Custom List – Specifies the number of a LIST function block which will hold additional configuration parameters. See <i>Step 3. Set Up the Custom List</i> later in this example.
iiComPort	INT input	Communication Port – For serial Modbus only: Identifies the port to be used for communication with Modbus slaves: 1 = COM1; 2 = COM2; etc.
iiSlaveAddress	INT input	For Serial Modbus: Slave Address : This parameter specifies the address of the Modbus slave. Slave addresses may range from 1 to 247. A slave address of 0 is used to generate a broadcast message. Read functions (Function Type 1 through 5) are not valid for a broadcast message. For Open Modbus (TCP/IP) Unit Number – A value which specifies the unit number of the slave device to match the transactions against. This unit number is included in the Modbus / TCP message prefix.
idiTimeout	DINT input	Timeout - This parameter specifies, in milliseconds, the amount of time the system waits for a response message from the Modbus slave. If you enter 0, a default value equivalent to 3000 milliseconds will be used. Otherwise, the value can range up to 65535 milliseconds.
isIPAddress	STRING input	IP Address of MODBUS Slave - For Open Modbus only: Specifies the Open Modbus slave device's IP address, such as 120.0.0.13.
odiStatus	DINT output	Status - This output parameter reports a status / error code which indicates the status of the CUSTOM function block execution. This parameter is only updated at the completion of the CUSTOM function block execution. For a complete description of error and status codes for all ACCOLIII function blocks, see the on-line help.

Parameter Name	Type of Variable	Description
oudDoneCount	UDINT output	Done Count – This is incremented by 1 whenever a Modbus communication message has been read/written and processed. This parameter is only updated at the completion of the CUSTOM function block execution. To determine whether or not the communication transaction was successful, check the Status.
obDoneFlag	BOOL output	Done Flag – This is set to TRUE whenever a Modbus communication message has been read/written and processed. This parameter is only updated at the completion of the CUSTOM function block execution. To determine whether or not the communication transaction was successful, check the Status.

Step 3.Set up the Custom List

The custom list defines various additional parameters which govern Modbus Master operation. Insert a LIST20 function block, and configure its entries as follows:

Note

Variables in the list are automatically interpreted as type INT. This allows other types (REAL, DINT, SINT) to be used without the user needing to perform type conversions.

Parameter	Variable type (recommended)	Description																				
iiListnumber	INT	List Number – This entry must match the iiCustomlist parameter value on the CUSTOM function block.																				
ianyElement1 (Required)	INT	<p>Function Code - This entry specifies the MODBUS Function Code. In some cases the Function code requires an offset of 1000 to make the transmitted address correct –see Coil/Register Address. The entries below show the numbers to enter for a particular function:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Function</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Read Coil Status</td> </tr> <tr> <td>2</td> <td>Read Input Status</td> </tr> <tr> <td>3</td> <td>Read Holding Registers</td> </tr> <tr> <td>4</td> <td>Read Input Registers</td> </tr> <tr> <td>5</td> <td>Force Single Coil</td> </tr> <tr> <td>6</td> <td>Preset Single Register</td> </tr> <tr> <td>7</td> <td>Read Exception Status</td> </tr> <tr> <td>8</td> <td>Force Multiple Coils</td> </tr> <tr> <td>9</td> <td>Preset Multiple Registers</td> </tr> </tbody> </table>	Value	Function	1	Read Coil Status	2	Read Input Status	3	Read Holding Registers	4	Read Input Registers	5	Force Single Coil	6	Preset Single Register	7	Read Exception Status	8	Force Multiple Coils	9	Preset Multiple Registers
Value	Function																					
1	Read Coil Status																					
2	Read Input Status																					
3	Read Holding Registers																					
4	Read Input Registers																					
5	Force Single Coil																					
6	Preset Single Register																					
7	Read Exception Status																					
8	Force Multiple Coils																					
9	Preset Multiple Registers																					
ianyElement2 (Required)	INT	Coil / Register Address – This entry specifies the starting address for coil, input, or register operations. The address transmitted to the Slave will be one less than the value specified here unless the Function code has an offset of 1000. For example, the address 7031 will be sent as 7030 for Function code 3, and 7031 for Function code 1003.																				
ianyElement3	INT	Data Count – A value specifying the number of coils, inputs, or																				

Parameter	Variable type (recommended)	Description
(Required)		registers to be read. The value can range from 1 to 2000 for coils and inputs, 1 to 125 for 16-bit registers and 1 to 62 for 32-bit registers.
ianyElement4 (Required)	INT	I/O List Number – A value specifying the list which will hold the coil or register data to be sent or received.
ianyElement5 (Required)	INT	I/O Array – RESERVED FOR FUTURE USE. Even though it is not used, it must be configured with a variable, which is set to 0.
ianyElement6 (Required)	INT	Data Size – This value specifies the data size to be used when accessing Holding Registers via function codes 3, 6, and 9. This option does not affect the operation of any other function codes. If signal 6 is unwired a default value is 3 is used. Valid values are: 1 = use single bit register data 2 = use 8-bit register data 3 = use 16-bit integer register data (default) 4 = use 32-bit integer register data with each double integer value taking up one register address. 5 = use 32-bit floating point data with each floating point value taking up one register address. In this mode, the selection is also used on Preset Single Register and Preset Multiple Register commands. 6 = Use 32-bit integer register data with each double integer value taking up two register addresses. 7 = Use 32-bit floating point data, with each floating point value taking up 2 register addresses. In this mode, the selection is also used on Preset Single Register and Preset Multiple Register commands.
ianyElement7 (Optional)	INT	Bit Order - Determines the ordering of bits in a byte. Note: Unless users have some special application requiring a different bit order, this value should always be left at the default of 0. If set to 0 (default), bit order within a byte is as follows: bit0 = lowest order coil : bit7 = highest order coil. If set to 1, bit order within a byte is as follows: bit0 = highest order coil : bit7 = lowest order coil.
ianyElement8 (Optional)	INT	Byte Order - Determines the order in which bytes are transmitted. Note: Unless users have some special application requiring a different byte order, this value should always be left at the default of 1. If set to 1 (default) the high order byte is transmitted first. If set to 0, the low order byte is transmitted first.
ianyElement9 (Optional)	INT	Word Order - Determines the order in which words are transmitted. Note: Unless users have some special application requiring a different word order, this value should always be left at the default of 1. If set to 1 (default) the high order word is transmitted first.

Parameter	Variable type (recommended)	Description
		If set to 0, the low order word is transmitted first.
ianyElement10 (Optional)	INT	RTS/CTS Delay Time – This is an optional element used only in serial MODBUS applications. It specifies a time delay. The time delay can either be used to monitor for CTS being raised, or to delay transmitting a message; the choice of how it is used is specified by the RTS/CTS Delay Mode. The delay value can range from 0 milliseconds to 65,535 milliseconds.
ianyElement11 (Optional)	INT	RTS/CTS Delay Mode - This is an optional value which specifies how the RTS/CTS Delay Time is used. There are two choices for the mode. 0 = <i>Message Transmit Delay Mode</i> : After RTS is raised, a delay timer will be started. (The length of the delay is determined by the value of RTS/CTS Delay Time). No message will be sent until after this delay has expired. The value of CTS does not affect the operation of this mode. <i>NOTE: In order for this mode to work, RTS-CTS must be jumpered or the CTS must be received before the specified delay expires.</i> 1 = <i>Monitor For CTS Mode</i> : After RTS is raised, the time delay will be used as the maximum time to wait within which CTS must be received. If CTS is received at any time before this delay expires, the message transmission begins. If CTS is NOT received prior to the expiration of the delay, no response will be sent, and an error will be reported.
odiStatus	DINT	Status Codes – Status or error codes for this list. See on-line help for details.

Step 4. Set up the I/O List

At this point, all that remains is to set up the I/O List which will hold the coil/register values sent to and received from the MODBUS slave device.

The list identification number must match the **I/O List Number** entry in the Custom List, and the length of the list must be greater than or equal to the number of variables needed to hold the coil/register data.

Type conversions will automatically be performed on the coil/register data to ensure that there is no conflict with the defined variables in the I/O List.

Configuring Your ControlWave Controller as a Modbus Slave or Enron Modbus Slave Device

Your ControlWave controller can be configured to serve as a Modbus Slave or Enron Modbus Slave device. The process is similar to that described in this previous example, i.e. configuring a CUSTOM function block, defining a CUSTOM list, etc. Full instructions about how to perform this configuration for Modbus Slave applications is included in the online help of ControlWave Designer.

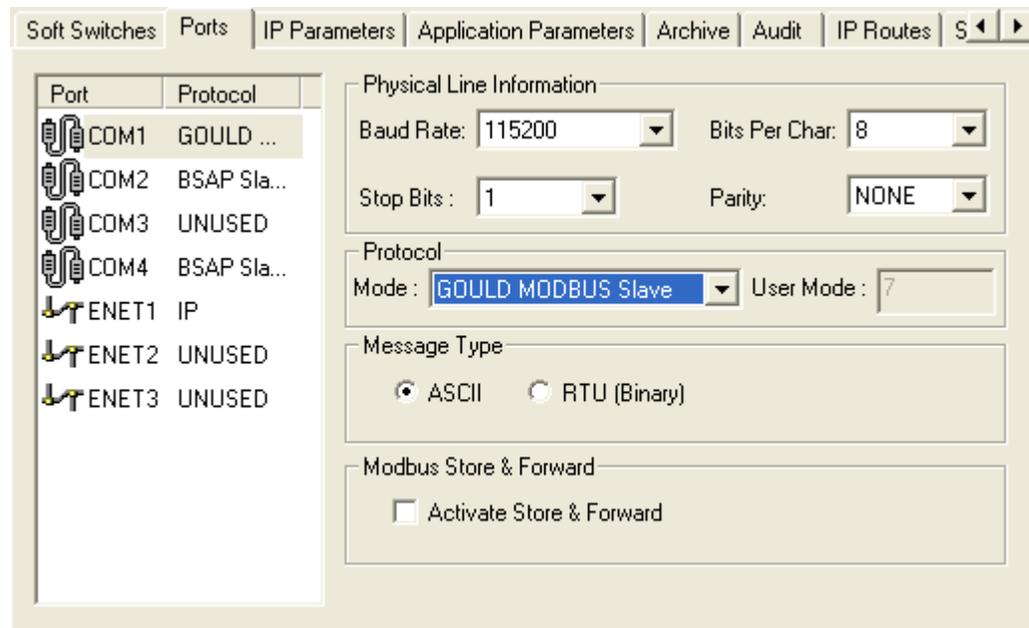
Modbus Ports

Configuring Modbus Ports

To use Modbus protocol to communicate you must configure a Modbus port. Any ControlWave-series controller's serial port can be configured for Modbus communications.

With communications active in LocalView or NetView, *right-click* on the ControlWave icon, and choose **RTU→RTU Configuration Parameters** from the pop-up menus.

The Flash Configuration Utility opens. Click on the 'Ports' tab.



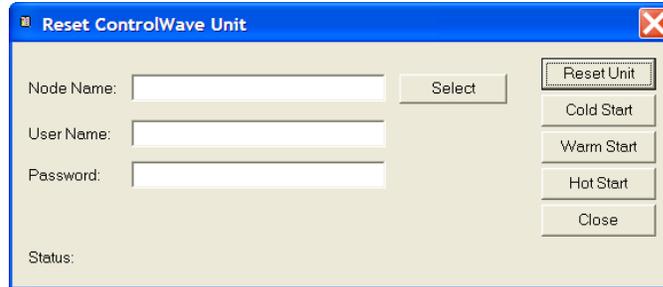
Mode	Sets the operational mode for this ControlWave-series controller in the Modbus network. Valid values are Modbus Master, ENRON Slave, or GOULD Modbus Slave, depending upon the controller's role. Alternatively, you can select USER_MODE for certain user-defined Modbus protocols.
Baud Rate	Indicates the baud rate used by this Modbus protocol. The default value for baud rate is 9600.
Bits Per Char	Indicates the number of bits used in a character. If the "Message Type" is 'ASCII', the number of bits is 7 or 8. The default is 8. If the "Message Type" is 'RTU (binary)' this parameter is fixed at 8 bits.
Stop Bits	Indicates the number of stop bits per character. The default is 1.
Parity	Specifies either ODD, EVEN, or NONE for the parity. The default is

	NONE.
Message Type	Specifies the type of data to be transmitted. Valid values are RTU (binary) or ASCII.
Modbus Store & Forward	Activates the Modbus Store and Forward feature (only for Modbus slaves). Modbus store and forward accepts a message with a given slave address, replaces that address with a <i>different</i> address, and then retransmits the message. See the ControlWave Designer online help for more information on configuring the Modbus store and forward feature
Activate Store & Forward	

Reset ControlWave Utility

The Reset ControlWave Utility allows you to power cycle the ControlWave (turn it off and then back on). This operation stops the project currently running in the ControlWave, reboots the ControlWave, clears its dynamic memory, and runs internal startup checks.

You can also restart the ControlWave project in “cold”, “warm,” or “hot” mode.



To reset a ControlWave:

1. Start the utility by clicking **Start → Programs → OpenBSI Tools → Debugging Tools → Reset ControlWave**.
2. Click [**Select**] to choose the node name of the controller you want to reset.
3. Sign on to the unit by entering a valid “**User Name**” and “**Password**” combination, then click [**Reset Unit**].
4. If signed on successfully, a prompt appears asking you to confirm that you want to proceed with the reset. Click [**OK**] to proceed or [**Cancel**] to abort the operation.
5. Wait for the “**Status**” field to report completion of the reboot operation.
6. Click [**Close**] to exit the utility.

To re-start a ControlWave project:

1. Start the utility by clicking **Start → Programs → OpenBSI Tools → Debugging Tools → Reset ControlWave**.
2. Click the [**Select**] button to choose the node name of the controller running the project.
3. Sign on to the ControlWave by entering a valid “**User Name**” and “**Password**” combination.
4. Choose one of the following options:
 - [Cold Start]** Click to re-start the ControlWave project from the beginning, using default values for all variables.
 - [Warm Start]** Click to re-start the project from the beginning of the task cycle, with saved values used for all RETAIN variables. (Only possible if Static RAM preserved; i.e. not cleared by battery failure or switch setting.)

[Hot Start] Click to stop the project, and then re-start the project from the beginning of the task cycle.

5. Wait for the **"Status"** field to report "Restart Complete."
6. Click **[Close]** to exit the utility.

Running RESETCW From the Command Line

You can also run the RESETCW program from the command line. The syntax for this is as follows: `RESETCW node_name command username password`

where: *node_name* is the name of the ControlWave unit to be reset

command is either 'RESET' to reset the unit, or a startup mode, as described, above, which could be either 'COLD', 'WARM', or 'HOT'.

username password is a valid username password combination for this node.

Example: `RESETCW CW3 COLD ROB 123ABCDE`

Security

There are two methods available for configuring ControlWave security:

- Configure security one RTU at a time using the Security page of the Flash Configuration utility. See *Chapter 5* of the *OpenBSI Utilities Manual* (part number D301414X012) for information on doing this.
- Use the Security Management Tool (available in OpenBSI 5.8 and newer). The Security Management Tool is very similar to the first method, except it allows you to send the same user security configuration information to all active RTUs in the network, thereby reducing the amount of work to configure your security system.

Note

To use the Security Management Tool, NetView must be running, and you must be logged on in NetView as the system administrator. This requires you to log on as the SYSTEM user with the appropriate password. Also, a SYSTEM user with matching password must exist at each RTU.

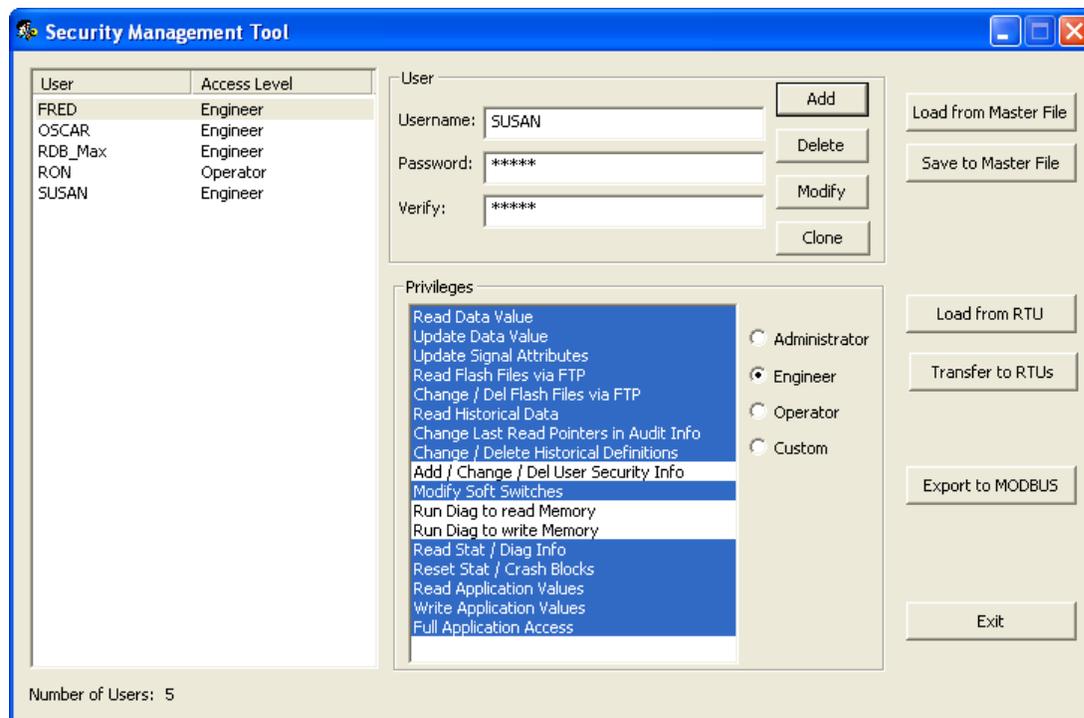
Starting the Security Management Tool

Click as follows:

Start → Programs → OpenBSI Tools → ControlWave Tools → User Management Tool

Defining Usernames and Passwords, and Specifying Privileges

The Security Management Tool allows you to create usernames and passwords for ControlWave users, and to define user privileges. In this way, you can create restrictions on who has access to various features and functions of the controller.



Adding A New User

A ControlWave-series controller supports up to 240 different users (previous to OpenBSI 5.8, 32 users was the maximum allowed). To add a new user, first enter the user's name (up to 16 characters long) in the **Username** field, and enter a password (up to 16 characters long) for that user in both the **Password** and **Verify** fields. (The password does not appear as you type it.)

Important

Prior to OpenBSI 5.8 Service Pack 1, some OpenBSI programs which communicated with the controller (such as DataView) only supported shorter usernames and passwords (10 characters or less for the username, 6 characters or less for the password) and UPPERCASE only for the password. If you have any OpenBSI workstations you haven't upgraded yet, you may want to limit ControlWave usernames/passwords to conform to this. Newer OpenBSI software conforms to the 16 character ControlWave limits.

Note

ControlWave firmware versions prior to version 5.20 only support the previous limit of 32 users. In those cases, the Security Management Tool only stores the first 32 users in your user list at the RTU. If RDB_MAX and/or the currently logged on user are not among the first 32 users in your user list, the Security Management Tool accommodates this by replacing the 32nd and, if necessary the 31st users in the list with RDB_MAX and/or the currently logged on user.

To select the privileges for this user, click **Custom** and then select the individual privileges in the **Privileges** list box, to highlight them. Alternatively, you can choose **Operator**, **Engineer**, or **Administrator** for a particular user, which automatically highlights privileges associated with those user categories. The tables, on the next page, show the privileges associated with these user categories, and list what the various privileges mean.

When all desired privileges have been selected, click **Add** to add the user to the system.

Note

Every ControlWave-series controller has a special user called **RDB_Max**. This user account defines the *maximum* privileges allowed for RDB protocol messages coming into the ControlWave. (Programs such as DataView, the Harvester, etc. use RDB to request data from the ControlWave.) You can neither delete nor rename the RDB_Max user, but you *can* change its privileges.

The table below shows the privileges associated with the Operator, Engineer, and Administrator categories:

Privilege	Operator	Engineer	Administrator
Read Data Value	✓	✓	✓
Update Data Value	✓	✓	✓
Update Signal Attributes		✓	✓
Read Flash Files via FTP		✓	✓
Change / Del Flash Files via FTP		✓	✓
Read Historical Data	✓	✓	✓
Change Last Read Pointers in Audit Info		✓	✓
Change / Delete Historical Definitions		✓	✓
Add / Change / Del User Security Info			✓
Modify Soft Switches		✓	✓
Run Diag to read Memory			✓
Run Diag to write Memory			✓
Read Stat / Diag Info	✓	✓	✓
Reset Stat / Crash Blocks	✓	✓	✓
Read Application Values		✓	✓
Write Application Values		✓	✓
Full Application Access		✓	✓

The table, below, describes the meaning of each privilege:

Privilege	Description
Read Data Value	Allows this user to read data values from this controller.
Update Data Value	Allows this user to change data values in this controller.
Update Signal Attributes	Allows this user to modify the status of inhibit / enable flags in the controller.
Read Flash Files via FTP	Allows this user read access (via File Transfer Protocol) to files stored in this ControlWave's Flash memory. This could include the ControlWave boot project, source files

Privilege	Description
	(* .ZWT), etc.
Change / Del Flash Files via FTP	Allows this user (via File Transfer Protocol) to add, change or delete files stored in the ControlWave's Flash memory. This could include the ControlWave boot project, source files (* .ZWT), etc.
Read Historical Data	Allows this user to view historical data (Audit / Archive information) from the controller, either via web pages, or OpenBSI DataView.
Change Last Read Pointers in Audit Info	Allows the user to delete Audit Trail data from the controller.
Change / Delete Historical Definitions	Allows this user to add, change or delete historical definitions via the Flash Configuration Utility.
Add / Change / Del User Security Info	Allows this user to add, change, or delete security configuration information via the Flash Configuration Utility.
Modify Soft Switches	Allows this user to change Soft Switch values on the soft switches page of the Flash Configuration Utility.
Run Diag to read Memory	Allows this user to run diagnostics to read memory at the controller.
Run Diag to write Memory	Allows this user to run diagnostics to write to memory at the controller.
Read Stat / Diag Info	Allows this user to view communication statistics and other information on the Statistics web pages.
Reset Stat / Crash Blocks	Allows this user to reset statistics and crash block areas on the Statistics web pages.
Read Application Values	Allows this user to read values using the ControlWave Designer OPC Server.
Write Application Values	Allows this user to modify values using the ControlWave Designer OPC Server.
Full Application Access	Allows this user full privileges to perform debugging operations in ControlWave Designer.

Loading the Security Definition for a Particular RTU into the tool

To load the security configuration information from a particular RTU into the tool, click **Load from RTU** and select the RTU from the Select New Node dialog box. Depending on the number of users defined, it may take some time for the load to complete.

Modifying the Privileges of an Existing User

To change the privileges of an existing user, select the user's name from the list of **Usernames** and select / de-select privileges for that user in the **Privileges** list box. When you finish making selections, click **Modify** to store the modified privileges for that user.

You can only modify privileges for users defined as **Custom**. The privileges for operators, engineers, administrators, etc. are fixed.

Cloning an Existing User

If you want to create several users with identical privileges, click on the name of the user that has the desired privileges, and then click **Clone**.



Type the number of users you want to create in the **Number of Cloned Users** box, then press the **[Enter]** key. Users named CLONE will appear. You can then modify those users with new usernames and passwords. (OpenBSI 5.8 and newer.)

Deleting an Existing User

To delete a user from the system, select the user's name from the **Usernames** list and click **Delete**.

Note

You **cannot** delete the RDB_Max user and you **cannot** delete the current user or any user who is **currently signed on** to this ControlWave.

Saving / Retrieving the Master Security Configuration File

You can export the security configuration information for the users on this ControlWave to an encrypted binary file called the **master file**.

To export a master security configuration file, click **Save to Master File**.

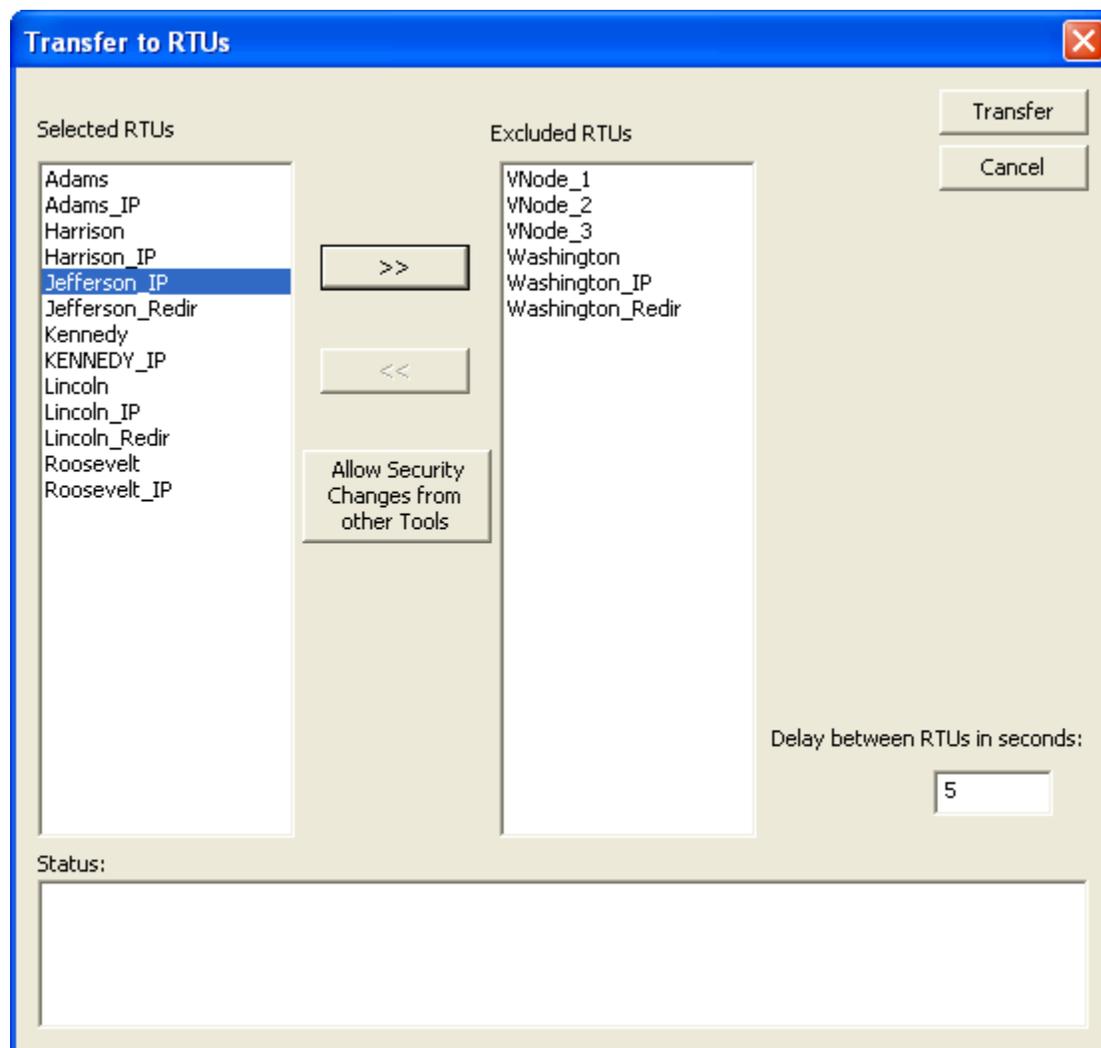
To import a master security configuration file, click **Load from Master File**.

The import/export feature requires OpenBSI 5.8 and newer.

Sending the Security Configuration to One or More RTUs

Once you define all the privileges for the users, you can transfer them to the controller. If you want to define the same users for multiple controllers, you can specify to which controllers you want to send the configuration.

Click the **Transfer to RTUs** button to open the Transfer to RTUs dialog box.



When the dialog box opens, all RTUs appear in the **Selected RTUs** list box.

If you don't want to send the security configuration to all RTUs, hold down the **[Ctrl]** key and then click on any RTUs you want to exclude, then click the **[>>]** button.

To select a whole block of RTUs in the list press **[Ctrl]** to select the first RTU, then press the **[Shift]** key with the cursor on the last RTU in the block.

You can click the **[<<]** button to move RTUs back from the **Excluded RTUs** list.

When the **Selected RTUs** list box shows only the RTUs you want to transfer security information to, click the **Transfer** button. The tool sequentially sends the security configuration to each RTU in the **Selected RTUs** list.

Removing the Lockout That Prevents Other Tools From Modifying Security Configurations

If you want to allow other tools such as the Flash Configuration Utility to modify the security configuration of one or more RTUs, select the RTUs for which you want to allow this so they show in the **Selected RTUs** list box, then click **Allow Security Changes from**

other Tools and then click **Transfer**. The tool sequentially removes the lockout from each RTU in the **Selected RTUs** list.

Exporting Security Data for Modbus Devices

To export security data for use in MODBUS registers, click **Export to MODBUS** and specify the location for the file.

Default Switch Settings at the Controller

The default switch on each RTU (also called the Use/Ignore Soft Switches) should be turned ON (Use Soft Switches), when you are finished, otherwise the special default security account (SYSTEM) remains active and the security configuration you store via the Security Management Tool is ignored.

- The default switch on the ControlWave Process Automation Controller is SW1-3.
- The default switch on the ControlWave Low Power (LP) Controller is SW4-3.
- The default switch on the ControlWave MICRO Process Automation Controller is SW2-3.
- The default switch on the ControlWave Gas Flow Computer (GFC) is SW2-3.
- The default switch on the ControlWave Electronic Flow Meter (EFM) is SW2-3.
- This default switch on the ControlWave Explosive-Proof Flow Computer (XFC) is SW1-3.
- The default switch on the CW_10 and CW_30 is SW2-3.

Other Security-Related Issues

In addition to defining usernames and passwords, there are certain other issues to consider when setting up security for your network. Some of these issues are not strictly related to ControlWave, but to the entire process control network.

The most secure process control network would be *self-contained* (that is, it would no connections to the outside Internet). In addition, it would be *isolated* from other network activities in your organization. For example, the process control network would not be on the same network as the billing department. If, however, your organization chooses to share networks, or include connections to the outside world, security should be a paramount concern.

Security Protocols for PPP Communication (PAP, CHAP)

If you are using the Point-to-Point (PPP) Internet Protocol, via the ControlWave-series controller's serial ports, you should consider using either the PAP or CHAP security protocols. CHAP is considered more secure than PAP, because it uses encryption. For information on these protocols, see the *Security Protocols* section in this manual.

OpenBSI Security

Since ControlWave-series controllers are designed to be used in OpenBSI networks, OpenBSI security should also be configured. For information on configuring security for your OpenBSI network, please see Chapter 6 of the *OpenBSI Utilities Manual* (part number D301414X012). This chapter also includes information on granting proxy access to the network, from remote OpenBSI Workstations, which can have security implications.

Network Infrastructure (UDP and TCP Sockets)

While this is generally more concerned with network traffic issues, UDP and TCP sockets throughout an IP network must match for each controller and PC Workstation. It is recommended that these numbers be changed from their default values, for greater security. For information on setting these at the OpenBSI level, see the *OpenBSI Utilities Manual* (part number D301414X012). For information on setting these in your ControlWave controller, please see the *IP Parameters* section of this manual.

Security Configuration for your Human Machine Interface (HMI) Software

Most supervisory control and data acquisition (SCADA) systems incorporate various levels of security. If you are using OpenEnterprise, security can be set up for both the OpenEnterprise Server database, individual database objects, and the OpenEnterprise Workstation(s). See the OpenEnterprise online help for details.

Virus Protection for Your Workstations

Each workstation should be equipped with virus protection software and you should subscribe to a regular update service for this software, so the virus protection remains current as new viruses become a threat. Virus protection is particularly important if the workstation has *any* connection to outside networks.

Firewall Software for Your Networks

If your process control network is connected to the Internet, you are *strongly* urged to purchase and configure commercially available firewall software, to provide some level of protection against external intrusion from hackers.

Physical Security

Consider issues related to the physical security of your workstations, and the nature of your operation. Are the workstations in a *high traffic* area? Are they in a room which can be locked up to prevent unauthorized access? Are the environmental conditions (excessive dust, high/low humidity) in the room poor?

For ControlWave-series controller, have you removed the RUN/REMOTE/LOCAL keys from the controllers and put them in a safe place? Otherwise, someone could accidentally or deliberately change the controller's operating mode.

Networked Surveillance of Remote Sites using ControlWave

If surveillance cameras are part of your physical security scheme, you should be aware that ControlWave-series controllers have successfully been used as part of surveillance schemes. Using third-party security cameras and software, captured images can be uploaded to the controller, and stored in flash memory. Customers can download the images via File Transfer Protocol (FTP). See the *ControlWave Security Vision Application User's Guide* (part number D301427X012) for more information.

Maintain Current Backups

This is valuable not only for security issues, but for any type of disaster recovery. System administrators should back up all necessary files on a regular basis, and store the backup media (tapes, CDs, DVDs, USB drives) in a safe, secure location, preferably off-site.

Human Factors

This may seem basic, but the downfall of security is often the human factor. No matter how well you configure your security system, if you post your passwords right next to the workstation or write them down where unauthorized persons can read them, or if you neglect to change default passwords or choose passwords that are simple for an intruder to figure out (like your name or the name of the company) then the protection provided by your security system can be severely diminished.

Security Protocols (CHAP and PAP)

Security Protocols (CHAP and PAP) Used on PPP Links

While it is not required to implement security on PPP links, users should be aware of the possibility of unauthorized access to their networks by an intruder (hacker), and strongly consider using one of the two supported security protocols.

Two standard protocols have been implemented for security on PPP links in networks of ControlWave RTUs: Challenge Handshaking Authentication Protocol (**CHAP**) and Password Authentication Protocol (**PAP**). These protocols operate in a client/server arrangement. Typically, CHAP should be used since it is more secure.

The CHAP (or PAP) *server* would be a ControlWave-series controller; the CHAP (or PAP) *client* could be either a ControlWave-series controller, or an OpenBSI workstation.

Whether a workstation or controller is the client, the client must always supply a valid username/password combination in order to gain access to the server.

If the OpenBSI workstation were the client, the username and password would be entered directly by the user in response to a login prompt. These must match one of the username / password combinations stored in the ControlWave.

If a ControlWave controller is the client, one of its valid usernames must be entered in the "**Challenge Protocol Default Username**" field in the **Ports** tab of the Flash Configuration Utility. This username / password text string for that username will automatically be transmitted in response to a login prompt from the server.

At the user-level, both of these security methods are similar. The difference occurs in the underlying *operation* of the protocols.

Challenge Handshaking Authentication Protocol (CHAP)

The CHAP server (ControlWave) issues an encrypted **challenge message** (which appears as a normal login prompt) to any CHAP client (workstation or ControlWave controller) requesting access. The supplied username and password will be encrypted according to a pre-defined secret encryption key; the result is called the **response message**.

Even though the username / password combination for a particular user does NOT change on each login attempt, the encrypted challenge and response messages is *different* on each attempt. This helps prevent an intruder from replicating the proper response message for a given challenge message, either through trial and error or through *brute force* searches of all possible challenge messages. Another characteristic of CHAP is that even after the client has logged in, subsequent challenge / response transactions will occur to verify that the connection is still with a valid user.

CHAP Example 1: In the first example of using CHAP, the CHAP client is a PC workstation, and the CHAP server is a ControlWave controller:

CHALLENGE HANDSHAKING AUTHENTICATION PROTOCOL (CHAP)

EXAMPLE 1 - WORKSTATION TO CONTROLLER

- 1 Login prompt (Challenge) message arrives at CHAP client workstation, and is decrypted using a secret key. To the user, it appears as a normal login prompt. User enters username and password combination.

CHAP client

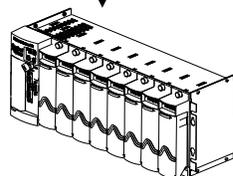


Username: JOHN
Password: smartguy

- 2 Username / Password combination is encrypted using secret key and sent out (Response Message). NOTE: This may involve a 2 message interchange between the client and server.

Each login attempt from a particular node results in a different encrypted response message, therefore, anyone intercepting the message after encryption would not be able to re-use it to gain access.

- 3 CHAP Server controller decrypts the response message using the secret key. The result is a username / password combination.



CHAP server

- 4 Server checks its password database to verify that the received username/password combination is valid.

Is "JOHN" and 'smartguy' a valid username and password combination? If YES, grant access, otherwise deny access.

p5092kjfdkdhgfls83172kdfisa

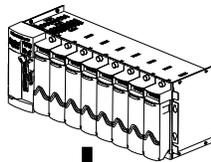
CHAP Example 2: The second example using CHAP is very similar, except in this case, the CHAP client is *another* ControlWave controller. For this reason, the username/password combination (default IP user) must be stored as parameters in flash memory and referenced by the Challenge Protocol Default Username.

CHALLENGE HANDSHAKING AUTHENTICATION PROTOCOL (CHAP)

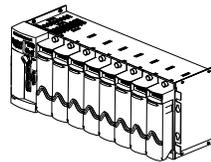
EXAMPLE 2 - CONTROLLER TO CONTROLLER

- 1 Username / password combinations are stored in the ControlWave as FLASH parameters. The choice of which user is the Default IP User is specified through the **"Challenge Protocol Default Username"** entered on the IP Parameters page.
- 2 Login prompt (Challenge) message arrives at CHAP client, and is decrypted using a secret key.

CHAP Client



OqiuH4wrtpojfi;rt rt[lqerfiq



CHAP Server

- 3 Username / Password combination (default IP user string) is encrypted using secret key and sent out (Response Message). NOTE: This may involve a 2 message interchange between the client and server.

Each login attempt from a particular node results in a different encrypted response message, therefore, anyone intercepting the message after encryption would not be able to re-use it to gain access.

- 4 CHAP Server controller decrypts the response message using the secret key. The result is a username / password combination.
- 5 Server checks its password database to verify that the received username/password combination is valid. If it is, grant access, otherwise, deny access.

Password Authentication Protocol (PAP)

PAP requires a client requesting access to provide a username and password, similar to CHAP. PAP is a simpler method of protection, however, that has certain characteristics which make it *less secure* than CHAP.

PAP allows passwords to be sent as clear plain text unencrypted strings of characters. There is a possibility, therefore, that an unauthorized person could intercept a password message, and then subsequently use the password to gain access.

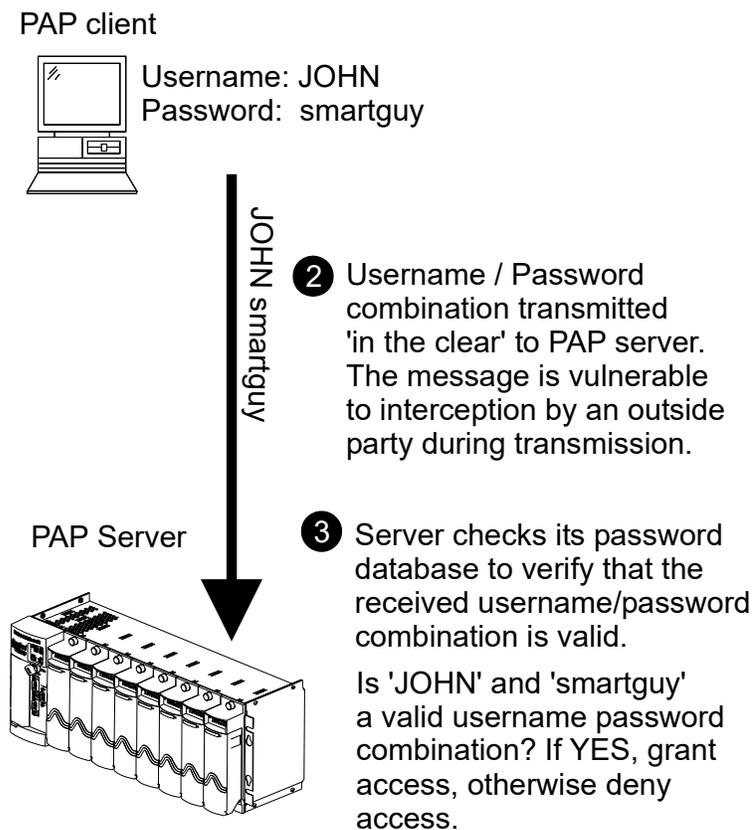
PAP also has no safeguards against repeated attempts to log in. For example, someone using trial and error to guess a password, or someone using software which performs a brute force search of all possible passwords could gain access.

PAP Example 1: In the first example of using PAP, the PAP client is a PC workstation, and the PAP server is a ControlWave controller.

PASSWORD AUTHENTICATION PROTOCOL (PAP)

EXAMPLE 1 - WORKSTATION TO CONTROLLER

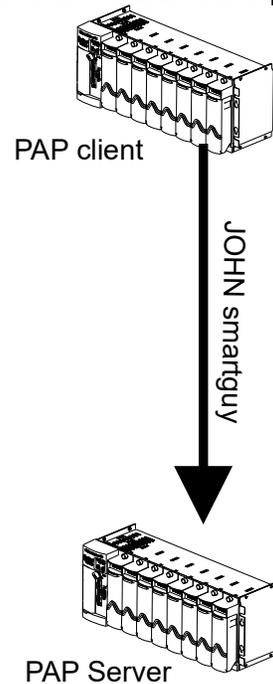
- 1 User logs in at a client workstation



PAP Example 2: The second example using PAP is very similar, except in this case, the PAP client is *another* ControlWave controller. For this reason, the username/password combination must be stored as a parameter in flash memory and designated using the Challenge Protocol Default Username.

PASSWORD AUTHENTICATION PROTOCOL (PAP) EXAMPLE 2 - CONTROLLER TO CONTROLLER

- 1 Username / password combinations are stored in the ControlWave as FLASH parameters. The choice of which user is the Default IP User is specified through the **"Challenge Protocol Default Username"** entered on the IP Parameters page.



- 2 In response to a login prompt, Username / Password combination for default IP user is transmitted 'in the clear' to PAP server. The message is vulnerable to interception by an outside party during transmission.

- 3 Server checks its password database to verify that the received username/password combination is valid.

Is 'JOHN' and 'smartguy' a valid username password combination? If YES, grant access, otherwise deny access.

References:

Lloyd, Brian , and Simpson, William, "PPP Authentication Protocols", Daydreamer Computer Systems Consulting Services, RFC 1334, October, 1992. (available at www.ietf.org)

Rivest, Ronald, "The MD5 Message-Digest Algorithm", MIT Laboratory for Computer Science, and RSA Data Security Inc., RFC 1321, April, 1992. (available at www.ietf.org)

Security Protocols (DNP3 SAV5)

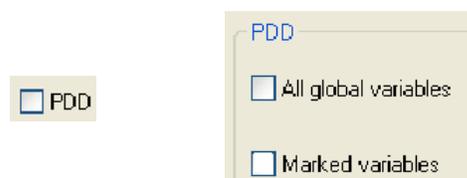
The industry-standard Distributed Network Protocol 3 (DNP3) supports Secure Authentication version 5 (SAV5). ControlWave firmware 6.10 supports DNP3 SAV5 for ControlWave PAC, ControlWave Micro, and ControlWave Express devices.

To implement SAV5 in the ControlWave platform, you must create an application that includes the required structures for DNP3 communications and secure authentication.

ControlWave Designer online help includes topics that describe how to configure DNP3 and SAV5. ControlWave Designer includes a sample application as a starting point. Once you configure the application, download it, and store the DNP3 key used for SAV5 there are other things you need to consider:

SAV5 Implementation Considerations

- **Limit Communication Port Usage** - The secure authentication offered by DNP3 SAV5 only exists on communication ports that solely use DNP3 SAV5. If you have other ports on the ControlWave device that communicate via Modbus, BSAP, or some other protocol, those communications will **not** have secure authentication. Therefore, to maintain security, you should only use DNP3 SAV5 on the device; you should **not** use other protocols on other ports, except for temporary local physical connections, such as when you are downloading the application (see the next two items).
- **Disable Remote Data Modifications via OpenBSI** – De-select any variables in your application marked for inclusion in the Process Data Directory (PDD) and clear all PDD check boxes in the Resource Settings dialog box.



DNP3 makes no use of the PDD and leaving variables in the PDD would allow BSAP access.

- **Require Physical-Access-Only for non-SCADA Communications-** OpenBSI/ControlWave Designer connections to the device do **not** support secure authentication. Therefore, once your application is complete, tested, and downloaded, you should lock-down the device to prevent download of an application either from ControlWave Designer or from the OpenBSI 1131 Downloader. Communication via ControlWave Designer should only occur when you have direct, physical access to the device. Presumably you would also limit direct physical access by mounting the device in a locked control room, or locked tamper-resistant enclosure.

- **SCADA System Conformance with DNP3 SAv5.** The SCADA system you use to communicate with the ControlWave must support DNP3 SAv5 and must have the key for the device.

Locking Down the ControlWave Device and Application

- The ControlWave device should only reside on a private network. Placing the device on the same network as your business systems, or on networks with computers that access the World Wide Web, offer opportunities for a security breach.
- For enhanced data security when using an IP/Ethernet connection or a serial connection into an IP terminal server, Emerson Energy and Transportation Solutions recommends adding an industrial router with VPN and firewall security. Recommended solutions include the MOXA EDR-810, the Hirschman Eagle One, or the Phoenix mGuard rs4000 (or equivalents). An example of how to install one of these devices for the RTU can be found in the Emerson Energy and Transportation Solutions [MOXA® Industrial Secure Router Installation Guide](#) (part number D301766X012). For further information, contact your Emerson Impact Partner or the individual vendor's website.
- Before you download the complete, tested, ControlWave Designer application locally, set system variable `_APPLICATION_LOCKED` to TRUE.
- After you have downloaded the complete, tested, ControlWave Designer application, set keys (or mode switches) to disable remote application download via OpenBSI or ControlWave Designer.

ControlWave Process Automation Controller (PAC), ControlWave Micro models with key switch 	ControlWave Micro models with Mode Switch	ControlWave Express CPU/System Controller Board
On the PSSM module, turn the key to the RUN position, and then remove the key and keep it in a safe place. This prevents downloading an application using ControlWave Designer or OpenBSI,	On the PSSM module, Set SW1-1 (the upper DIP switch) to the Open (right) position and SW1-2 (the lower DIP switch) to the Closed (left) position to place the ControlWave Micro in Local mode, which is used for normal running operations.	Set CPU switch SW1-1 to the OFF position and SW1-2 to the ON position to place the RTU in Local mode, which is used for normal running operations.

- Set switches to disable remote upgrade of system firmware and to prevent changes to soft switches:

ControlWave Process Automation Controller	ControlWave Micro Process Controller	ControlWave Express Process Controller
Set CPU switch SW1-2 to OFF to lock soft switches. Set CPU switch SW3-2 to ON to disable remote system firmware upgrades.	Set CPU switch SW2-2 to OFF to lock soft switches. Set CPU switch SW2-6 to OFF to disable remote system firmware upgrades.	Set CPU switch SW2-2 to OFF to lock soft switches. Set CPU switch SW2-6 to OFF to disable remote system firmware upgrades.

System Tasks (Warm/Cold Starts)

A **system task** is a grouping of one or more programs which execute only under certain pre-defined conditions.

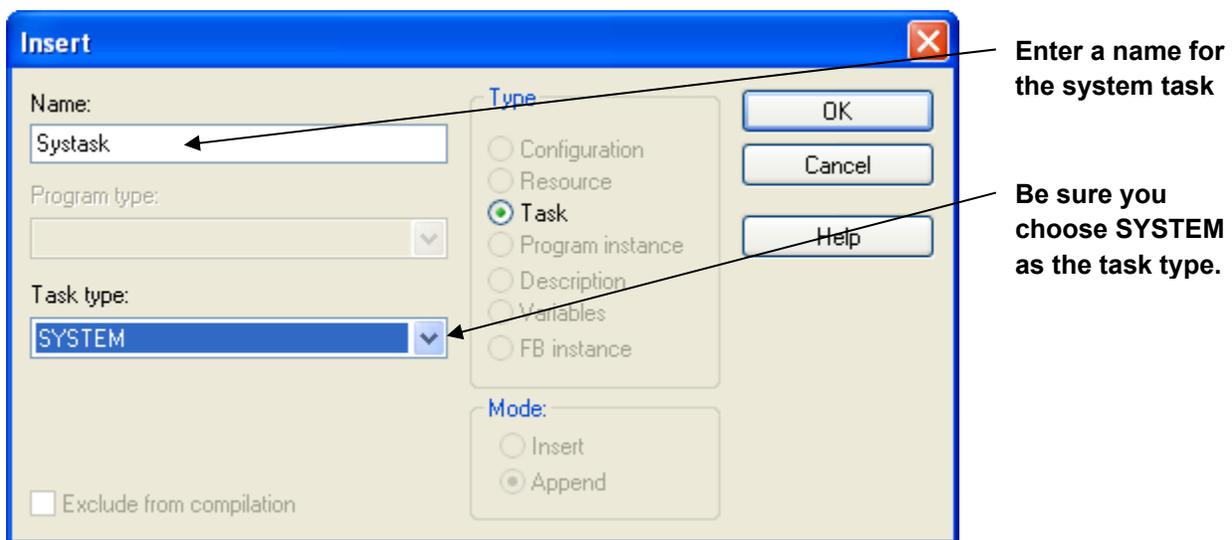
Creating a Warm start or Cold start System Task

Note

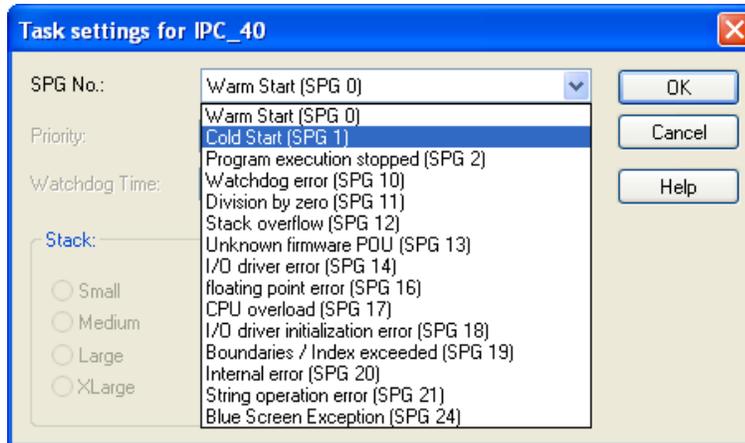
Refer to the section *Memory Usage* in this manual for an explanation of “warm start” and “cold start.”

When you define a warm start or a cold start system task, the programs in that task are only executed once at the application warm start or application cold start, respectively.

To create a warm start or cold start system task, create a task as you normally would, but be sure you choose 'SYSTEM' as the **"Task type"**.



After you click **OK**, you must choose either 'Warm Start (SPG 0)' or 'Cold Start (SPG 1)' as the condition which will cause this system task to be executed.



Click **OK** to finish.

Once your completed project has been downloaded into the ControlWave, it executes **only** when triggered by the specified condition.

System Variables

Important

With the exception of port configuration parameters, most users **do not** need to be concerned with these system variables. *Advanced* users, however, may find the system variables useful for many purposes, including:

- Monitoring the efficiency of task execution
 - Accessing the system date and time
 - Viewing diagnostic information about the communication ports
-

Various system variables are maintained by the ControlWave series controller. They are mapped to standard memory locations, and are accessible only when the user creates variables to specifically access those locations.

These variables are stored in static RAM and so are capable of surviving all application reloads and cold, warm, or hot application restarts or the CPU power off/on. They are set/modified by applications and remain that way until modified again. The only way these system variables lose their settings is when the SRAM Control switch setting forces a memory clear on a power on, the SRAM back-up battery is low, or if the system detects an SRAM corruption and reinitializes them. All input/output parameters are defined with their default settings when applicable. Other input parameters must be given some specific value by an application.

There are two ways to access the System Variables:

- Use the System Variable Wizard to create the variables, and for certain variables, enter values (this is the preferred method)
- Manually create variables according to the information contained in the System Variable Mapping Charts. You must use the name, data type and address shown in the charts.

Using the System Variable Wizard

1. Start ControlWave Designer.
2. Open your ControlWave project.
3. Start the System Variable Wizard by clicking **View**→**System Variable Wizard**.
4. The System Variable Wizard will appear.
 - You can access the various pages of the wizard by clicking on the different tabs.
 - Information on the use of individual variables is included in the System Variable charts (later in this section).

- In instances where there are too many variables to appear on a single page, a push button is included to call up a *different* page containing more variables. Simply click on the push button to access the new page.
- To create a variable, just check the appropriate box. The variable name shown is the name you must use in your ControlWave project to access the variable. That same name also should be used to look up a description of what the variable is used for in the System Variable Mapping Charts.
- Sometimes several variables will be grayed out until a *different* variable is created; this is because they are all related in some way.
- If the user may set a variable, an entry box or selection box will appear, with a default entry or selection - enter a new value in the entry box, if desired, or select from among the options presented.
- At the bottom of each page are check boxes which allow you to mark variables for “PDD” or “OPC” collection. NOTE: These boxes apply to ALL system variables; i.e. if you check PDD on one page, it applies to every system variable on every page. In addition, if you call up a system variable in the standard variable dialog box, and select one of these options, it will also apply to ALL system variables the next time you open the System Variable Wizard. You cannot pick and choose individual variables with this option.
- If you choose the “Write All” option (added in OpenBSI 5.4) ALL system variables will be created from every page of the wizard.
- Click [OK] to exit the System Variable Wizard, and save the changes you have made; otherwise, click [Cancel].

You can access the various pages of the System Variable Wizard by clicking on the different tabs.

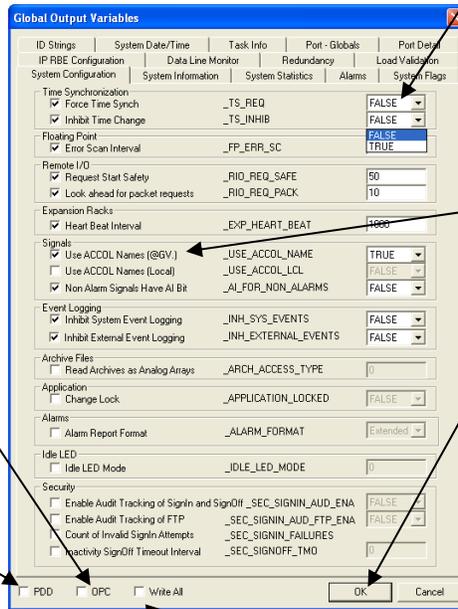
If the variable may be set by the user (input variable) choose from the selection box or enter a value in the entry box, as appropriate.

If you check "OPC" ALL system variables in this project will be marked for OPC collection.

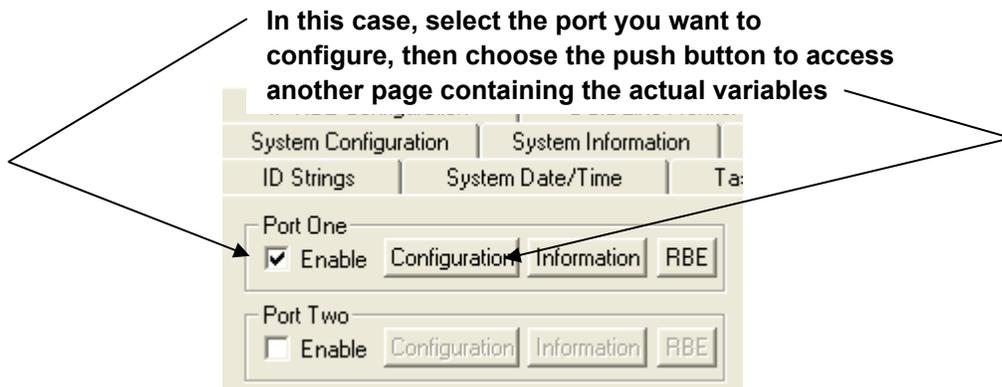
If you check "PDD" ALL system variables in this project will be marked for PDD collection.

To create a variable, check the box associated with it.

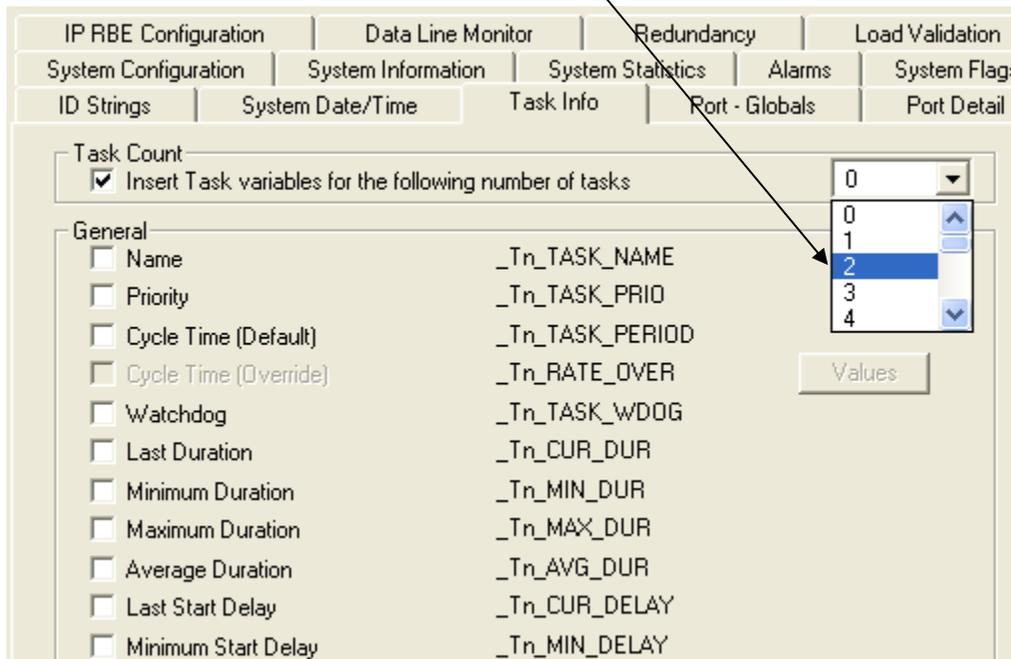
Only click on OK when you want to save all entries and exit the System Variable Wizard.



Select "Write All" only if you want to generate all the system variables on ALL the pages of the System Variable Wizard.



Select the number of tasks in your project; separate sets of the variables shown on this page are created for *each* task.



System Variable Mapping Charts

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
_PLCMODE_ON	%MX 1.0.0 : BOOL	1.0	<p>Status of the unit.</p> <p>TRUE = unit is powered on but an application is not loaded. (State field in the ControlWave Designer Project Control dialog box will also show ON.)</p> <p>FALSE = unit is power on and application is loaded.</p> <p>Only one of the variables _PLCMODE_ON, _PLCMODE_RUN, _PLCMODE_STOP, and _PLCMODE_HALT is set to TRUE at any given time.</p>
_PLCMODE_RUN	%MX 1.0.1 : BOOL	1.0	<p>Status of the ControlWave Designer application.</p> <p>TRUE = ControlWave project is running (State field in the ControlWave Designer Project Control dialog box will also show RUNNING.)</p> <p>FALSE = project is not running.</p> <p>Only one of the variables _PLCMODE_ON, _PLCMODE_RUN, _PLCMODE_STOP, and _PLCMODE_HALT is set to TRUE at any given time.</p>
_PLCMODE_STOP	%MX 1.0.2 : BOOL	1.0	<p>Status of the ControlWave Designer application execution.</p> <p>TRUE = ControlWave project is stopped. (State field in the ControlWave Designer Project Control dialog box will also show "Stop.")</p> <p>FALSE = ControlWave project is executing.</p> <p>Only one of the variables _PLCMODE_ON, _PLCMODE_RUN, _PLCMODE_STOP, and _PLCMODE_HALT is set to TRUE at any given time.</p>
_PLCMODE_HALT	%MX 1.0.3 : BOOL	1.0	<p>Debug status of the ControlWave Designer application.</p> <p>TRUE = ControlWave project is halted at a</p>

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
			breakpoint in debug mode (State field in the ControlWave Designer Project Control dialog box will also show "Halt (Debug)"). FALSE = project is not halted at a breakpoint.
_PLCDEBUG_BPSET	%MX 1.1.4 : BOOL	1.0	TRUE=one or more breakpoints are set for debugging purposes
_PLCDEBUG_FORCE	%MX 1.2.0 : BOOL	1.0	TRUE=one or more I/O variables are forced for debugging purposes
_PLCDEBUG_POWERFLOW	%MX 1.2.3 : BOOL	1.0	TRUE=power flow is active for Debugging purposes
_PLC_TICK_PER_SEC	%MW 1.44 : INT	1.0	Number of ticks per second (will be 250)
_PLC_SYS_TICK_CNT	%MD 1.52 : DINT	1.0	The number of ticks since the application was started.
_LOAD_BOOT_PRESENT	%MX 1.332.0 : BOOL	04.80	The boot project is present in FLASH memory.
_LOAD_SRC_PRESENT	%MX 1.333.0 : BOOL	04.80	The project source file (*.ZWT) is present in the FLASH memory.
_LOAD_MEM_PRESENT	%MX 1.334.0 : BOOL	04.80	There is a project loaded into memory (SDRAM) or SRAM, depending upon type of unit.
_LOAD_BOOT_CRC	%MD 1.336 DWORD	04.80	Cyclic redundancy (CRC) check number for boot project.
_LOAD_SRC_CRC	%MD 1.340 DWORD	04.80	Cyclic redundancy check (CRC) number for project source (*.ZWT).
_LOAD_MEM_CRC	%MD 1.344 DWORD	04.80	Cyclic redundancy check (CRC) number for project in memory.
_MAX_TASK	%MW 1.1000 : INT	1.0	The maximum number of user tasks supported.
_CUR_TASK	%MW 1.1002 : INT	1.0	The current number of user tasks.

Note

System variables which are specific to a particular task are preceded by a prefix. The first task system variable name begins with a '_T1' prefix, the second task system variable name begins with a '_T2' prefix, etc.

The next several system variables are created for the **first** Task in the system

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
_T1_TASK_NAME	%MB 1.1004 : SI_10	1.0	Array of 10 characters which defines the task's name.

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
_T1_TASK_PRIO	%MW 1.1014 : INT	1.0	The configured priority of this task.
_T1_TASK_PERIOD	%MW 1.1018 : INT	1.0	If the task is a cyclic task, this is the number of milliseconds between scheduled start times.
_T1_TASK_WDOG	%MW 1.1024 : INT	1.0	Configured time (in milliseconds) before which the task must complete (or an error will be generated)
_T1_CUR_DUR	%MW 1.1032 : INT	1.0	Number of ticks, which the task took to execute (from scheduled time to task completion)
_T1_MIN_DUR	%MW 1.1034 : INT	1.0	Minimum execution time for the task.
_T1_MAX_DUR	%MW 1.1036 : INT	1.0	Maximum execution time for the task.
T1_AVG_DUR	%MW 1.1038 : INT		Average execution time for the task.
_T1_CUR_DELAY	%MW 1.1040 : INT	1.0	Number of ticks after the scheduled time, which the current task has taken to start execution.
_T1_MIN_DELAY	%MW 1.1042 : INT	1.0	Minimum of CUR_DELAY since the system was started.
_T1_MAX_DELAY	%MW 1.1044 : INT	1.0	Maximum of delay since the system was started.
_T1_AVG_DELAY	%MW 1.1046 : INT	1.0	Average of delay since the system was started.

The next several system variables are created for the **second** Task in the system:

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
_T2_TASK_NAME	%M 1.1068 : SI_10	1.0	See above description for corresponding T1 variable
_T2_TASK_PRIO	%MW 1.1078 : INT	1.0	See above description for corresponding T1 variable
_T2_TASK_PERIOD	%MW 1.1082 : INT	1.0	See above description for corresponding T1 variable
_T2_TASK_WDOG	%MW 1.1088 : INT	1.0	See above description for corresponding T1 variable
_T2_CUR_DUR	%MW 1.1096 : INT	1.0	See above description for corresponding T1 variable
_T2_MIN_DUR	%MW 1.1098 : INT	1.0	See above description for corresponding T1 variable
_T2_MAX_DUR	%MW 1.1100 : INT	1.0	See above description for corresponding T1 variable
_T2_AVG_DUR	%MW 1.1102 : INT	1.0	See above description for corresponding T1 variable
_T2_CUR_DELAY	%MW 1.1104 : INT	1.0	See above description for corresponding

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
			T1 variable
_T2_MIN_DELAY	%MW 1.1106 : INT	1.0	See above description for corresponding T1 variable
_T2_MAX_DELAY	%MW 1.1108 : INT	1.0	See above description for corresponding T1 variable
_T2_AVG_DELAY	%MW 1.1110 : INT	1.0	See above description for corresponding T1 variable
<i>System variables for tasks would continue from this point on, following the same pattern as shown for the first and second tasks, above:</i>	<i>3rd task begins at %M1.1132 4th task begins at %M1.1196 5th task begins at %M1.1260 etc.</i>		

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
_LOAD_BOOT_PRESENT	%MX 1.332.0 : BOOL	04.80	The boot project is present in FLASH memory.
_LOAD_SRC_PRESENT	%MX 1.333.0 : BOOL	04.80	The project source file (*.ZWT) is present in the FLASH memory.
_LOAD_MEM_PRESET	%MX 1.334.0 : BOOL	04.80	There is a project loaded into memory (SDRAM) or SRAM, depending upon type of unit.
_LOAD_BOOT_CRC	%MD 1.336 DWORD	04.80	Cyclic redundancy (CRC) check number for boot project.
_LOAD_SRC_CRC	%MD 1.340 DWORD	04.80	Cyclic redundancy check (CRC) number for project source (*.ZWT).
_LOAD_MEM_CRC	%MD 1.344 DWORD	04.80	Cyclic redundancy check (CRC) number for project in memory.
_POWER_UP	%MX 3.0.0 : BOOL	1.0	Set on system restart. Cleared by the user if detection of power-fail recovery is needed.
_QUEST_DATE	%MX 3.0.1 : BOOL	1.0	Should be FALSE during normal operation. Set TRUE if the real time clock value is invalid and the unit needs a time synchronization message from the host. This occurs immediately after power-up, or after the project is downloaded; once the time synch message from the host is processed, this is set to FALSE. _QUEST_DATE also becomes TRUE if the SRAM backup battery voltage falls below

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
			the level necessary to power the real time clock (battery failed and must be replaced).
_TS_REQ	%MX 3.0.2 : BOOL := FALSE	1.0	Set to force time synch from master. Will be cleared when time synch is received.
_TS_INHIB	%MX 3.0.3 : BOOL := FALSE	1.0	Set if user wishes to inhibit processing of time synch messages.
_OCTIME_ERROR	%MX 3.1.0 : BOOL	4.50	Reports if time was changed on last time synch.
_TIME_000	%MD 3.4 : DWORD	1.0	Combination Julian Date and seconds since midnight. Used to time-stamp historical information.
_TIME_008	%MW 3.4 : INT	1.0	Julian Date.
_JULIAN_TIME	%MD 3.4 REAL	Not tied to firmware. Requires OpenBSI 5.7 Service Pack 1 or newer.	Julian Date and Time in REAL format. number of seconds since midnight. This is the floating point representation of the _TIME_000 value. The applications may use this value directly as the Julian time. The hex representation of this Julian time is used as an input in producing the date and time. Example: The date/time 1/18/2008 09:20 will be reported as 3.5435290E-019. The Hex representation of this floating point value is 20D12C4C. The first 2-byte value is the number of 4 second intervals since midnight. It is used to calculate time in HH:MM;SS where the seconds resolution, is in multiples of 4 seconds. Therefore, 20D1 represents 09:20:04. The second two byte value is the number of days since midnight of 12/31/1976. Therefore, 2C4C represents the date of 01/18/2008 .
_TIME_4SEC	%MW 3.6 : INT	1.0	Number of 4-second intervals since midnight.
_TIME_001	%MD 3.8 : DINT	1.0	Number of seconds since midnight.
_TIME_002	%MW 3.12 : INT	1.0	Year (4 digit)
_TIME_003	%MB 3.14 : SINT	1.0	Month (1 to 12)
_DAY_OF_WEEK	%MB 3.15 : SINT	1.0	Day within the current week (1=Sunday to 7=Saturday)
_TIME_004	%MB 3.16 : SINT	1.0	Date (1 to 31)
_TIME_005	%MB 3.17 : SINT	1.0	Hour (0 to 23)
_TIME_006	%MB 3.18 : SINT	1.0	Minute (0 to 59)

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
_TIME_007	%MB 3.19 : SINT	1.0	Second (0 to 59)
_POWER_OFF	%MD 3.20 : DINT	1.0	Number of seconds the system was powered off (after a power-fail recovery)
_CPU_OVERLD	%MD 3.24 : DINT	1.0	Number of times that the System Idle time was detected to be too small
_RIO_REQ_SAFE	%MD 3.28 : DINT := 50	1.0	Number of milliseconds before calculated due time to start collection of I/O from remote devices
_RIO_REQ_PACK	%MD 3.32 : DINT := 10	1.0	In milliseconds. If two RIO requests are to be made to the same IP address within this time, they are packed into a single TCP packet.
_BAD_ANY_CONST	%MD 3.36 : DINT	1.0	Invalid constants loaded onto variable of type 'ia' or 'iany'.
_BAT_OK	%MX 3.40.0 : BOOL	1.0	Should be TRUE during normal operation. It is set TRUE if ControlWave's SRAM backup battery is OK. If the battery cannot provide the necessary voltage to backup the SRAM (battery voltage low, battery failed, or missing) this becomes FALSE, and the battery should be replaced. It also shows FALSE if the backup battery is disabled by the jumper. On power-up after battery failure, all retain values in SRAM are set to 0, or their initial values. NOTE: Does not work for RTU 3340 - always will show a '1'
_HOT_CARD_IN_PROG	%MX 3.41.0 : BOOL	1.0	Set when ControlWave 'hot' card replacement is in progress. When set, I/O values for the board are held constant.
_USE_ACCOL_NAME	%MX 3.42.0 : BOOL	4.00	Used in response to RDB messages from OpenBSI. When set to TRUE global variable names (those beginning with @GV) will be translated to ACCOL signal name format before being sent back. If _USE_ACCOL_LCL, variables with instance names other than '@GV' will also be translated to ACCOL II signal format. Variable names must not exceed 20 characters, and must follow the 8 character basename, 6 character extension, and 4 character attribute rules of ACCOL II.
_AI_FOR_NON_ALMS	%MX 3.42.1 : BOOL	4.10	Alarm variables are only those variables which have been configured as alarms using one of the Alarm function blocks. A

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
			<p>non-alarm variable is any variable which hasn't been configured as an alarm using one of the Alarm function blocks.</p> <p>When <code>_AI_FOR_NON_ALMS</code> is set to FALSE, the Alarm Inhibit/Enable bit is permanently set to AI (alarm inhibit) for all non-alarm variables. (This is the default).</p> <p>When <code>_AI_FOR_NON_ALMS</code> is set to TRUE, the Alarm Inhibit/Enable bit for each individual non-alarm variable can be set by the user to either Alarm Inhibit (FALSE state), or Alarm Enable (TRUE state). Because these are NOT alarms, this has no practical effect, unless the AI / AE bit is used for some specific purpose, e.g. in the OpenEnterprise Database.</p>
<code>_INH_SYS_EVENTS</code>	%MX 3.42.2 BOOL	4.20	Certain system events are logged by the AUDIT system (for example, the power on of the unit, etc.) To inhibit logging of these events, set this to TRUE.
<code>_USE_ACCOL_LCL</code>	%MX 3.42.4 BOOL	4.90	If set to TRUE, local signals with instance names other than '@GV' will be converted to ACCOL II style signal names in the Signal Extractor's SIG file, by changing underscores '_' to periods '.', and system signal underscores to pound '#' signs.
<code>_SEC_SIGNIN_AUD_ENA</code>	%MX 3.42.5 BOOL	5.20	When set TRUE, enables audit logging of user sign-on and sign-off activities at the RTU. This bit once turned on will not turn off except on an application COLD start. To keep this enabled a cold start task should turn this back on. Default: FALSE
<code>_SEC_SIGNIN_AUD_FTP_ENA</code>	%MX 3.42.6 BOOL	5.20	When set TRUE, enables audit logging of FTP sign-on and sign-off activities at the RTU. Users can sign-on via FTP to look at the contents of the RTU flash memory area. Default: FALSE. Note: The RTU ignores the value of this system variable if <code>_SEC_SIGNIN_AUD_ENA</code> is FALSE.
<code>_INH_EXTERNAL_EVENTS</code>	%MX 3.42.7 BOOL	5.11	When set TRUE, turns off logging of user events such as calibration events, notes events, and clear history events.
<code>_ALARM_FORMAT</code>	%MB 3.43 SINT	4.60	Alarm Report Format. Default = FALSE –

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
			Long Format. For short format, set to TRUE.
_FLASH_FREE	%MD 3.44 : DINT	1.0	Common free FLASH space in the FLASH memory area.
_FL_FILE_USAGE	%MD 3.48 : DINT	1.0	Total FLASH memory used for files.
_FL_FILE_FREE	%MD 3.52 : DINT	1.0	Free FLASH memory space available for files.
_FL_HIST_USAGE	%MD 3.56 : DINT	1.0	Total FLASH memory used for Historical data.
_FL_HIST_FREE	%MD 3.60 : DINT	1.0	Free FLASH memory space available for Historical data.
_HOT_CARD_COUNT	%MD 3.64 : DINT	1.0	Hot card insert/remove count. Incremented at start of lock.
_FP_ERR_SC	%MD 3.68 : DINT := 60000	1.0	Scan interval for floating point errors in tasks. All floating point errors occurring during the scan interval are counted as a single error in the ControlWave. In the RTU 3340, all floating point errors occurring during the scan interval are registered individually. Default scan interval is 60000 (1 minute).
_SUSP_PERCENT	%MW 3.72 : INT	2.0	A measurement of the percentage of time that application-level tasks are suspended.
_EXP_HEART_BEAT	%MD 3.76 DINT :=1000	3.10	This is the rate (in milliseconds) at which the ControlWave issues a heartbeat poll message to the I/O Expansion Rack(s). Default is 1000 (1 second).
_TOTAL_ALARMS	%MD 3.80 UDINT	4.00	This value maintains a count of alarms generated. It is incremented by 1 each time a new alarm comes in. Users can reset the count as desired.
_TOTAL_AUD_EVENTS	%MD 3.84 UDINT	4.00	This value maintains a count of events logged by the Audit system. It is incremented by 1 each time a new event occurs. Users can reset the count as desired.
_TOTAL_AUD_ALARMS	%MD 3.88 UDINT	4.00	This value maintains a count of alarms logged by the Audit system. It is incremented by 1 each time a new alarm comes in. Users can reset the count as desired.
_ALARMS_PRESENT	%MX 3.92.0 BOOL	4.00	This variable is set to TRUE, whenever a new alarm has been generated since the variable was last cleared (set to FALSE). If the user wants to use this as a notification that new alarms have arrived, the user

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
			must include logic to turn it OFF (set it to FALSE) after receiving each notification, so that it will be ready when the next alarm comes in.
_AUD_EVT_PRESENT	%MX 3.93.0 BOOL	4.00	This variable is set to TRUE, whenever a new audit event has been generated since the variable was last cleared (set to FALSE). If the user wants to use this as a notification that a new audit event has arrived, the user must include logic to turn it OFF (set it to FALSE) after receiving each notification, so that it will be ready when the next audit event comes in.
_AUD_ALM_PRESENT	%MX 3.94.0 BOOL	4.00	This variable is set to TRUE, whenever a new audit alarm has been generated since the variable was last cleared (set to FALSE). If the user wants to use this as a notification that new audit alarms have arrived, the user must include logic to turn it OFF (set it to FALSE) after receiving each notification, so that it will be ready when the next audit alarm comes in.
_IDLE_LED_MODE	%MB 3.95 SINT	5.00	Specifies behavior of the IDLE LED on ControlWave XFC platform only. Possible mode values are: 0 = To conserve power, LED remains OFF for the first 58 seconds of every minute; in the final two seconds, acts as in Mode 2. 1 = To conserve power, LED is unused and is always OFF. 2 = LED reflects busy/idle status of the CPU. When ON, the CPU has idle time. When OFF, CPU is busy. Non-XFC platforms: LED always reflects busy/idle status of the CPU. When ON, the CPU has idle time. When OFF, CPU is busy.
_ALARMS_IBP_DEST1	%MX 3.96.0 BOOL	4.20	Alarms are present, that are to be shipped to Alarm Destination 1.
_ALARMS_IBP_DEST2	%MX 3.96.1 BOOL	4.20	Alarms are present, that are to be shipped to Alarm Destination 2.
_ALARMS_IBP_DEST3	%MX 3.96.2 BOOL	4.20	Alarms are present, that are to be shipped to Alarm Destination 3.
_ALARMS_IBP_DEST4	%MX 3.96.3 BOOL	4.20	Alarms are present, that are to be shipped to Alarm Destination 4.

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
_ALARMS_BSAP_PORT1	%MX 3.96.4 BOOL	4.20	Alarms are present, that are to be sent out BSAP port 1 to the host.
_ALARMS_BSAP_PORT2	%MX 3.96.5 BOOL	4.20	Alarms are present, that are to be sent out BSAP port 2 to the host.
_ALARMS_BSAP_PORT3	%MX 3.96.6 BOOL	4.20	Alarms are present, that are to be sent out BSAP port 3 to the host.
_ALARMS_BSAP_PORT4	%MX 3.96.7 BOOL	4.20	Alarms are present, that are to be sent out BSAP port 4 to the host.
_ALARMS_BSAP_PORT5	%MX 3.97.0 BOOL	4.20	Alarms are present, that are to be sent out BSAP port 5 to the host.
_ALARMS_BSAP_PORT6	%MX 3.97.1 BOOL	4.20	Alarms are present, that are to be sent out BSAP port 6 to the host.
_ALARMS_BSAP_PORT7	%MX 3.97.2 BOOL	4.20	Alarms are present, that are to be sent out BSAP port 7 to the host.
_ALARMS_BSAP_PORT8	%MX 3.97.3 BOOL	4.20	Alarms are present, that are to be sent out BSAP port 8 to the host.
_ALARMS_BSAP_PORT9	%MX 3.97.4 BOOL	4.20	Alarms are present, that are to be sent out BSAP port 9 to the host.
_ALARMS_BSAP_PORT10	%MX 3.97.5 BOOL	4.20	Alarms are present, that are to be sent out BSAP port 10 to the host.
_ALARMS_BSAP_PORT11	%MX 3.97.6 BOOL	4.20	Alarms are present, that are to be sent out BSAP port 11 to the host.
_LOCAL_ADDRESS	%MW 3.98 : INT	4.20	The BSAP local address programmed into the unit.
_EBSAP_GROUP	%MW 3.99 : INT	4.20	The EBSAP Group number programmed into the unit. A value of 0 indicates that standard BSAP is used (no EBSAP).
_CPU_BUSY_P1	%MW 3.100 : INT	4.20	This measures (in units of 0.1%) how occupied the CPU is, running application-level tasks, or active system tasks.
_ARCH_ACCESS_TYPE	%MD 3.102 : SINT	4.41	When non-zero, reads archive files as if they were analog arrays.
_APPLICATION_LOCKED	%MX 3.103.0 : BOOL	4.50	When set TRUE, prevents external control changes to project via ControlWave Designer. Also prevents project downloads.
_NHP_ADDITIONAL_MASK	%MD 3.104 :DWORD	4.20	This defines an IP mask for additional NHP addresses.
_HEAP_CUR_FREE	%MD 3.108 : DINT	4.40	Displays the current amount of RAM that is free for system use.
_HEAP_BLK_FREE	%MD 3.112 : DINT	4.40	Displays the size of the largest block of free volatile memory. (Either SDRAM or SRAM, depending upon the ControlWave platform being used.)
_HEAP_START_FREE	%MD 3.116 : DINT	4.40	The value of HEAP_CUR_FREE when the application (project) was started.

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
_HEAP_RBLK_FREE	%MD 3.120 : DINT	4.40	Displays the size of the maximum block of retain memory that is currently free.
_TS_DELTA_ACCURACY	%MD 3.124 : DINT	4.50	Turn-around time before accepting timestamp.
_SEC_SIGNIN_FAILURES	%MD 3.128 UDINT	5.20	Count of the number of failed sign-in attempts at the RTU.
_SEC_SIGNOFF_TMO	%MD 3.132 UDINT	5.20	Specifies a period of time (in seconds) after which the RTU automatically logs a user off for inactivity. Range: 0-86400
_FLASH_SECTOR_FAIL	%MD 3.136 [ARRAY 1..8 OF DWORD]	6.02	Displays a bit mask of sectors of retained flash memory that have failed.
_DISABLE_FATAL_FLASH_FAIL	%MX 3.168.0 BOOL	6.02	Set to determine how the RTU responds when it detects a flash memory chip failure: <p>FALSE = The ControlWave resets and status LEDs report a flash failure. The ControlWave CPU module must be replaced. (default)</p> <p>TRUE = The ControlWave status LEDs report a flash failure. The ControlWave prevents a reset and logs the failure. It continues to operate but data is no longer written to the failed flash memory chip, disabling the current historical log. Replace the ControlWave's CPU module as soon as possible.</p>
_FLASH_FAILURE	%MX 3.168.1 BOOL	6.02	Reports detection of a flash memory chip failure which may cause loss of historical data: <p>TRUE = Flash memory chip failure detected. PSSM LEDs 1, 2, and 3 turn ON. System will respond according to value of _DISABLE_FATAL_FLASH_FAIL system variable. ControlWave RTU's CPU module must be replaced</p>

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
			before returning it to service. FALSE = No failure detected. (default)

Per Task Parameters (generated according to the number of tasks specified to the SysVar wizard)

The next several system variables are generated for each individual task. The first task system variable name begins with a '_T1' prefix, the second task system variable name begins with a '_T2' prefix, etc.

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
_T1_SLIP	%MD 3.1032 : DINT	1.0	Number of times that the watchdog timer expired for the current task.
_T1_FP_ERR	%MD 3.1036 : DINT	1.0	Number of floating point errors detected (underflow, overflow, etc.)
_T2_SLIP	%MD 3.1064 : DINT	1.0	See above description for corresponding T1 variable
_T2_FP_ERR	%MD 3.1068 : DINT	1.0	See above description for corresponding T1 variable
<i>System variables for tasks would continue from this point on, following the same pattern as shown for the first and second tasks, above:</i>	<i>3rd task 3.1096 to 3.127 4th task 3.128 to 3.159 etc. etc.</i>		

Communication Statistics per port. A fixed 68-byte long statistics area is reserved per port. SysVar Wizard selects from the following parameters according to the port type.

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
Port 1 offsets 3000-3067:			
_P1_TYPE	%MB 3.3000 : SINT	1.0	0 = UNUSED 1 = BSAP Slave 2 = BSAP Master 15 = Custom Protocol 28 = Serial IP (PPP)
_P1_MODE	%MB 3.3001 : SINT	1.0	protocol mode
_P1_RECEIVES	%MD 3.3004 : DINT :=	1.0	This is the total number of messages received

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
	0		through this port. (Does not include protocol messages.)
_P1_TRANSMIT	%MD 3.3008 : DINT := 0	1.0	This is the total number of messages transmitted through this port. For the Master Port this includes poll messages as well.
_P1_POLLS _P1_RESP_TMO	%MD 3.3012 : DINT := 0	1.0 (Polls) 2.0 (Resp timeout)	For the Slave: This is the number of poll messages received by the Slave Port (as applicable). For the Master: Response Timeouts.
_P1_DISC_TRANS _P1_CONS_TMO	%MD 3.3016 : DINT := 0	2.0	For the Slave: Discards for transmission. If non-zero, may indicate _Px_POLL_PER value for the Slave Port should be a larger number. For the Master: Consecutive timeouts.
_P1_NAKS _P1_NAKS_RCV	%MD 3.3020 : DINT := 0	1.0 2.0	For the Slave: NAKs issued. For the Master: NAKs received.
_P1_CRC_ERR _P1_DISC	%MD 3.3024 : DINT := 0	1.0 2.0	For the Master: A message was received by the Master port with correct framing; however it failed the CRC check and was discarded. Usually, this is due to noise on the line. The Master ignores this message. For the Slave: A message which is received by the Slave but whose ACK is not received by the Master is retransmitted by the Master. The Slave discards the duplicate message and advises the Master by issuing an 'ACK, Message Discarded' response. This is commonly caused by noise on the line.
_P1_DISC_RCV	%MD 3.3028 : DINT := 0	2.0	Discarded ACKs received by the Master.
_P1_PROT_ERR	%MD 3.3032 : DINT := 0	2.0	This is the number of protocol errors for the Master port.
_P1_TMO_SEND	%MD 3.3036 : DINT := 0	2.0	This is the number of timeouts when sending messages.
Reserved	%Mx3.3040 - 3.3067		Reserved for future

:

Port 2 offsets 3068-3135: Replication of port 1 offsets.
Port 3 offsets 3136-3203: Replication of port 1 offsets.
Port 4 offsets 3204-3271: Replication of port 1 offsets.
Port 5 offsets 3272-3339: Replication of port 1 offsets.
Port 6 offsets 3340-3407: Replication of port 1 offsets.
Port 7 offsets 3408-3475: Replication of port 1 offsets.
Port 8 offsets 3476-3543: Replication of port 1 offsets.
Port 9 offsets 3544-3611: Replication of port 1 offsets.
Port 10 offsets 3612-3679: Replication of port 1 offsets.
Port 11 offsets 3680-3747: Replication of port 1 offsets.
Port 12 offsets 3748-3815: Replication of port 1 offsets.

Communication System Configuration Parameters

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
_ALM_RETRIES	%MW 3.4000 : UINT := 3	1.0	Number of times to retry alarm reports before slowing retry interval.
_ALM_RET_ACT	%MW 3.4002 : UINT := 60	1.0	Amount of time (in seconds) to wait before an alarm retry in ACTIVE state.
_ALM_RET_DEAD	%MW 3.4004 : UINT := 120	1.0	Amount of time (in seconds) to wait before an alarm retry in DEAD state.
_ETH_POLL_PER	%MW 3.4006 : UINT := 0	1.0	Timeout for traffic on Ethernet lines. (Ethernet poll period)
_ETH1_ACT	%MX 3.4008.0 : BOOL	1.0	Ethernet port 1 Active.
_ETH2_ACT	%MX 3.4008.1 : BOOL	1.0	Ethernet port 2 Active.
_ETH3_ACT	%MX 3.4008.2 : BOOL	1.0	Ethernet port 3 Active.
_NHP_IGNORE_NRT	%MX 3.4009.0 : BOOL	4.00	When set TRUE, node routing tables received via IP from the Network Host PC will be ignored.
_NHP_IGNORE_TS	%MX 3.4009.1 : BOOL	4.00	When set TRUE, time synchronization messages received via IP from the Network Host PC will be ignored.

Communication System – Port specific configuration parameters.

Port 1 - offsets 4010-4019: SysVar Wizard selects from the following parameters according to the port type.

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
_P1_POLL_PER	%MW 3.4010 : UINT := 5	1.0	Polling cycle time in seconds. Applies to Master and Slave. Master: Begins a new polling cycle on every timeout. If actual polling of all slave lasts longer than this timeout then the next polling cycle is started as soon as the current cycle finishes. Slave: If the master does not poll within this time period all messages queued to go up to the master are discarded.
_P1_WRITE_DEL	%MW 3.4012 : UINT := 0	1.0	Time in milliseconds. Applies to Master and Slave. Message reply delay. After CTS is received, wait this period before beginning the transmission.
_P1_RETRIES	%MW 3.4014 : UINT := 0	2.0	Number of link level retries. Applies to Master only. Default no retries, i.e. single transmission.

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
			For example, if set to 2, a total of 3 transmissions may occur.
_P1_ACTIVE	%MX 3.4016.0: BOOL	1.0	Output from comm. System. For BSAP Slave: One or more messages was received during the previous poll period. For BSAP Master: One or more slave nodes is still alive.
_P1_TS_FORCE	%MX 3.4016.1: BOOL := FALSE	1.0	Applies only to the Slave. Input/Output to/from comm. system. When set to TRUE a request asking for the TS/NRT message is sent to the port master.
_P1_DTR_STATE	%MX 3.4017.0 BOOL	4.20	Reports the current state of the Data Terminal Ready (DTR) line for this port. TRUE means DTR active.
_P1_DCD_STATE	%MX 3.4017.1 BOOL	4.20	Reports the current state of the Data Carrier Detect (DCD) line for this port. TRUE means DCD active.
_P1_DIAL_ACTIVE	%MX 3.4017.2 BOOL	3.11	When TRUE, indicates that dialing is in progress on this port.
_P1_TS_DIS	%MX 3.4018.0: BOOL := FALSE	1.0	Applies only to the Slave. Input to the comm. System. When set to TRUE do not process the TimeSync information at this port. Set it to FALSE to accept and process the TimeSync at this port (Default).
_P1_NRT_DIS	%MX 3.4018.1: BOOL := FALSE	2.0	Applies only to the Slave. Input to the comm. System. When set to TRUE do not process the NRT information at this port. Set it to FALSE to accept and process the NRT at this port (Default).
_P1_IDLE_POLL	%MX 3.4018.2: BOOL := FALSE	2.0	Applies only to the Master. Input to the comm. system. When set to FALSE the IDLE polling is Enabled (Default). Set this to TRUE to disable the IDLE polling.
_P1_ALM_DIS	%MX 3.4018.3: BOOL := FALSE	2.0	Input to the comm. System. When set to FALSE the port can transmit alarm reports (Default). Set this to TRUE when alarm reports are not to be transmitted through this port.
_P1_IMM_DIS	%MX 3.4018.4: BOOL := FALSE	3.10	Input to the comm. System. When FALSE (the default), this port can operate in immediate response mode. When set to TRUE, this port CANNOT operate in immediate response mode.
_P1_IGNORE_ECHO	%MX 3.4018.5 BOOL	3.10	On two-wire RS485 ports, anything sent is echoed back. Set to TRUE, to ignore the data echoed back.
_P1_DIAL_PORT	%MX 3.4018.6 BOOL	4.00	Set to TRUE to allow dialing on this port.
_P1_AUTO_DTR	%MX 3.4018.7 BOOL	4.20	The function of this variable varies depending

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
			upon the type of port. For all port types, DTR is disabled if DCD is inactive. In addition, the following restrictions apply: For BSAP Master Ports, DTR is only on while a communication transaction is in progress; once the transaction is over, DTR is dropped, until the completion of the poll period. For Custom Master ports, DTR is dropped after the transaction and no other request is pending. For Generic Serial Ports, no processing is done since DTR control is expected to be in manual mode.
_P1_LOCAL_PORT	%MX 3.4019.0 BOOL	4.20	Defines this port as a BSAP Local Port. It will answer polls to any BSAP address, even if this ControlWave has a different local address than the incoming poll message.
_P1_RESET_STATISTICS	%MX 3.4019.1 BOOL	5.30	Resets the statistics in the array for the I/O expansion rack. Used with the RIO 485 interface. See the ControlWave Designer online help for more information.
_P1_INH_BSAP_SLAVE	%MX 3.4019.2 BOOL	5.43	When set TRUE prevents BSAP slave communications on this port.
Port 2 - offsets 4020-4029 (replication of port 1 offsets as applicable).			
Port 3 - offsets 4030-4039 (replication of port 1 offsets as applicable).			
Port 4 - offsets 4040-4049 (replication of port 1 offsets as applicable).			
Port 5 - offsets 4050-4059 (replication of port 1 offsets as applicable)			
Port 6 - offsets 4060-4069 (replication of port 1 offsets as applicable)			
Port 7 - offsets 4070-4079 (replication of port 1 offsets as applicable)			
Port 8 - offsets 4080-4089 (replication of port 1 offsets as applicable)			
Port 9 - offsets 4090-4099 (replication of port 1 offsets as applicable)			
Port 10 - offsets 5000-5009 (replication of port 1 offsets as applicable)			
Port 11 - offsets 5010-5019 (replication of port 1 offsets as applicable)			
Port 12 - offsets 5020-5029 (replication of port 1 offsets as applicable)			
Other Communication parameters – Data Line Monitoring			

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
_DLM_PORT	%MB 3.5000 : SINT :=	1.0	1-based port number to monitor.
_DLM_R_PTR	%MD 3.5001 : USINT	1.0	Position in string to load next character.
_DLM_READ	%MB 3.5003 : STRING	1.0	Storage for string which has been read.
_DLM_WRITE	%MB 3.5088 : STRING	1.0	String which has been written.
String Data:			
_CW_NAME_STR	%MB 3.5400 : STRING	2.0	Defines the name.
_CW_DESCRIPTION_STR	%MB 3.5485 : STRING	2.0	Description: Automatically set by the system to the firmware ID.
_CW_CONTACT_STR	%MB 3.5570 : STRING	2.0	Defines the contact.
_CW_LOCATION_STR	%MB 3.5655 : STRING	2.0	Defines the location.
_CW_LOAD_STR	%MB 3.5740 : STRING	4.40	Defines the project version.
_S1_IO_BOARD_ID_STR	%MB 3.6000 : STRING	2.0	ID string for 1 st I/O board.
_S2_IO_BOARD_ID_STR	%MB 3.6085 : STRING	2.0	ID string for 2 nd I/O board.
_S3_IO_BOARD_ID_STR	%MB 3.6170 : STRING	2.0	ID string for 3 rd I/O board.
_S4_IO_BOARD_ID_STR	%MB 3.6255 : STRING	2.0	ID string for 4 th I/O board.
_S5_IO_BOARD_ID_STR	%MB 3.6340 : STRING	2.0	ID string for 5 th I/O board.
_S6_IO_BOARD_ID_STR	%MB 3.6425 : STRING	2.0	ID string for 6 th I/O board.
_S7_IO_BOARD_ID_STR	%MB 3.6510 : STRING	2.0	ID string for 7 th I/O board.
_S8_IO_BOARD_ID_STR	%MB 3.6595 : STRING	2.0	ID string for 8 th I/O board.
Redundancy parameters: (See <i>ControlWave Redundancy Setup Guide (D5123)</i> for details.)			
Additional Communication System configuration parameters.			
_SLAVE_PORT	%MW 3.8000 : UINT := 1	2.0	Port number for the network slave port. Valid values are: 1 - 11 (serial COM ports) 15 = IP (Ethernet or PPP)
_MSG_TIMEOUT	%MD 3.8004 : DINT := 30000	2.0	Timeout for all messages being tracked (all global messages passing through this node are tracked). If this value is <= 0 then the default time out of 30000 milliseconds (30 seconds) is assumed.
_NEW_NRT_RCVD	%MX 3.8008.0 : BOOL := FALSE	2.0	Output from Comm. system. TRUE indicates a new NRT was received and has to be propagated to the slave RTUs. FALSE following the TRUE state indicates that BSAP Master has or is in process of sending the NRT to all slave RTUs. Can be set to TRUE by the user to force transmission of NRT to slave nodes.

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
_SLAVE_DEAD	%M 3.8010 Map this to a user defined data type of a BOOL array with 127 elements. B_127 : ARRAY [1..127] OF BOOL; If this array is also to be manipulated using the Dataview or equivalent tool then the data type must be a two dimensional array with 127x1 or 1x127. B_1_1 : ARRAY [1..1] OF BOOL; B_127 : ARRAY [1..127] OF B_1_1; In addition, it must be registered with REG_ARRAY.	2.0	Applicable to the Master only. Array elements 1-127 are used by the system to report the current status of the slaves 1-127. When a node is not responding to the POLLS it is declared as dead and the corresponding status is set to TRUE. (NOTE: This logic can be reversed via _BSAP_FLAG_SENSE.) This array is similar to the #NODE.xxx. signals used in ACCOL II for Network 3000-series controllers.
_SLAVE_POLL_DIS	%M 3.8138 Map this to a user defined data type of a BOOL array with 127 elements. B_127 : ARRAY [1..127] OF BOOL; If this array is also to be manipulated using the Dataview or equivalent tool then the data type must be a two dimensional array with 127x1 or 1x127. B_1_1 : ARRAY [1..1] OF BOOL; B_127 : ARRAY [1..127] OF B_1_1; In addition, it must be registered with REG_ARRAY.	2.0	Applicable to the Master only. Array elements 1-127 are used as input to the comm. system to indicate whether the slave(s) are to be polled or not. When an array element is set to TRUE the corresponding node is not polled. (NOTE: This logic can be reversed via _BSAP_FLAG_SENSE.) This array is similar to the #NDARRAY used in ACCOL II for Network 3000-series controllers, except by default, the logic is opposite.
_BSAP_FLAG_SENSE	%MX 3.8265.0 BOOL :=FALSE	4.00	Default is FALSE. When set TRUE, reverses the interpretation of BOOL values in all four control and status arrays. See table, below:

Control or Status Array	BSAP_FLAG_SENSE=FALSE (Default)	BSAP_FLAG_SENSE=TRUE
_SLAVE_DEAD element	TRUE = Slave Node Dead; FALSE = Slave Node Alive	TRUE = Slave Node Alive; FALSE = Slave Node Dead
_SLAVE_POLL_DIS element	TRUE = Do NOT poll this node; FALSE = Poll this node	TRUE = Poll this node; FALSE = Do NOT Poll this node
_Px_DEAD_ARRAY element	TRUE = EBSAP Slave Node Dead; FALSE = EBSAP Slave Node Alive	TRUE = EBSAP Slave Node Alive; FALSE = EBSAP Slave Node Dead
_Px_DISABLE_ARRAY element	TRUE = Do NOT poll this EBSAP node; FALSE = Poll this EBSAP node	TRUE = Poll this EBSAP node; FALSE = Do NOT Poll this EBSAP node

Port 1 – offsets 8384-8428			
_P1_TIMEOUT	%MD 3.8384 : DINT := 500;	2.0	In milliseconds. Link level response timeout. This timeout value must be set approximately 50 ms higher than the facing slave's Message Write Delay.
_P1_CYCLE_INT	%MD 3.8388 : DINT	4.20	The interval (in milliseconds) at which DTR is enabled (Fast Duty Cycle).
_P1_CYCLE_TIMEO	%MD 3.8392 : DINT	4.20	Fast duty cycle timeout.
_P1_WRITE_TMO	%MD 3.8396 : DINT := 0	2.0	Common. In milliseconds. Since the CTS must be received in order to transmit, this time out is added to the expected message transmission time at the effective Baud Rate for this port. The response message must be completely transmitted before the resulting timeout. Default is dynamic and calculated based on the current baud rate.
_P1_LOW_SL	%MB 3.8400 : SINT := 0	2.0	Output parameter. Set to the value from the flash parameter. It is the lowest slave Address for this port.
_P1_HIGH_SL	%MB 3.8401 : SINT := 0	2.0	Output parameter. Set to the value from the flash parameter. It is the highest slave Address for this port.
_P1_VSAT_MIN_RESP	%MD 3.8402 : INT	4.40	The minimum period of time that a VSAT Slave Port will wait before responding to a message from the master. The main purpose of this delay is to allow a VSAT Master Port enough time to send messages to multiple slaves without getting a response from any of them until all of those messages

Port 1 – offsets 8384-8428			
			have been sent out.
_P1_VSAT_MAX_RESP	%MD 3.8404 : INT	4.40	<p>Beginning with ControlWave firmware 04.80, this variable has two possible uses:</p> <ol style="list-style-type: none"> 1. The maximum period of time (in milliseconds) that a VSAT Slave Port will wait before responding to a message from the master. This timer is meant to allow enough time for a data response to become available. One of the following can happen before this timer expires: a) An alarm message is sent. b) A data response message is sent. If the timer expires without either of these messages being sent, then a) A DOWN-ACK is sent (if received message was a data request) or b) An ACK/NO DATA message will be sent (if received message was a poll). 2. Beginning with ControlWave firmware 04.80, if the port is a BSAP Slave Port and immediate response is configured for this port, (i.e. _Pn_IMM_DIS is set to FALSE), this variable specifies how long the BSAP Slave driver must wait for a data response. If the response is received before this time, the response will be sent immediately. However, if the the timer expires then the driver will send a DOWN-ACK protocol response to the Master. This timeout value may range from 50 to 32767 milliseconds.
_P1_VSAT_UP_ACK_WAIT	%MD 3.8406 : INT	4.40	This is the period of time the VSAT Master will wait, after sending an UP-ACK-WITHOUT-POLL message (to terminate a poll transaction), to allow time for the VSAT Slave to get ready for the next request.
_P1_RBE_ERE_COUNT	%MD 3.8408 UDINT	4.40	Count of the number of Exception Reports (ER) generated and posted to the Report Pool for port <i>n</i> . This count is incremented each time a new exception is reported. Application may reset this count.
_P1_RBE_RM_COUNT	%MD 3.8412 UDINT	4.40	Count of the number of Report

Port 1 – offsets 8384-8428			
			Messages successfully transmitted for port n. Each RM can have more than 1 Exception Report in it. This count is not incremented when RM are retransmitted. Application may reset this count.
_P1_RBE_POOL_OVERFLOW	%MD 3.8416 UDINT	4.40	Count of the number of Exception Reports the RBE FB wanted to post in the Report Pool but could not because the Report Pool was full. These exceptions may essentially be lost for the RBE Client at this port. Application may reset this count.
_P1_RBE_POOL_SIZE	%MW 3.8420 UINT	4.40	The capacity of the Report Pool for port n. This size governs the maximum number of Exception Reports that can be outstanding at any given time without being successfully reported. This size is determined based on the expected rate of exception occurrence and the throughput at the port. This size can be changed on line. However, for the new size to be effective the RBE FB must be re-run with the ioablnit parameter in TRUE state. When this variable is set to 0 (default), the RBE is not active for the port. Adjust this parameter only if the Report Pool overflows or if the system wide RAM usage needs to be redistributed.
_P1_RBE_REXMIT_COUNT	%MD 3.8422 USINT	4.40	Indicates number of retransmissions of the last Report Message. This happens when an expected RBE_ACK is not received from the RBE Client. Following the RBE_ACK_TMO the last RM is retransmitted. On such retransmissions the RM_COUNT is not incremented. This count is automatically reset back to 0 whenever an ACK is received.
_P1_RBE_RM_SINCE_ACK	%MD 3.8423 USINT	4.40	Indicates number of Report Messages sent since last ACK was received. Count is adjusted whenever an ACK is received. This count is continuously incremented for each Report Message sent when the ACK Limit is set to 0.
_P1_RBE_CLIENT_ID	%MD 3.8424 USINT	4.40	Identity of the RBE Client for port n. (0xA3 = default) Valid range available = 0x00-0xFF. Only time this will require a

Port 1 – offsets 8384-8428			
			change is when the RBE Client in use has a different ID.
_P1_RBE_ERE_FORMAT	%MD 3.8425 USINT	4.40	Exception Report format type for port n.
_P1_RBE_ACK_LIMIT	%MD 3.8426 USINT	4.40	Indicates if and when an ACK is expected from the RBE Client. 0 = Never wait for an ACK for Report Message (default) n = 1-127 = send up to n Report Messages (RM) then stop if an ACK is not received for any of the Report Messages. When the reporting has stopped while waiting for an ACK, a timeout will result in the retransmission of the last RM. The periodic retransmission will occur until an ACK is received.
_P1_RBE_STOP_RPT_MSG	%MX 3.8427.0 : BOOL	4.40	FALSE = Start/Restart Report Message transmission at port n. TRUE = Temporarily stop sending Report Message for port n. When restarting from the stopped state the RM will begin with the next new Exception Report. This does not prevent new Exception Reports from being put into the buffer pool. It is possible to overflow the buffer pool while this variable is set.
_P1_RBE_PENDING	%MX 3.8427.1 : BOOL	4.40	This variable is set to TRUE when there is one or more Exception Reports pending for port n. It will automatically be changed to FALSE when the last Exception Report from the Report Pool has been successfully transmitted.
_P1_RBE_USE_ACCOL_NAME	%MX 3.8427.2 : BOOL	4.40	This variable determines the format of the variable name in the Exception Report. TRUE = the variable names are in ACCOLII format. FALSE = the variable names are in IEC61131 format.
_P1_RBE_GO_ACT_ON_START	%MX 3.8427.3 : BOOL	4.40	Variable determines the initial state for the RBE following an application COLD or WARM start. FALSE (default) = RBE is to send the WAITING_INIT message to the RBE Client and wait for the INIT_REQ message. Periodically, ACK_TMO, repeat the wait message until the init message is received. TRUE = Following a successful

Port 1 – offsets 8384-8428			
			initialization the RBE enters the active state where the database scan and exception reporting take place.
_P1_RBE_REPEAT_TIMEOUT	%MD 3.8428 DINT	4.40	Specifies the time to wait for an RBE ACK. If an ACK is not received within this period then retransmit the last Report Message. Periodically repeat this until an RBE ACK is received from the client. Ignore unrelated or out of sequence ACKs. A new Init or Demand Request will force an exit from this state. This timeout is also used during the initialization stage to determine when a WAITING INIT message will be re-sent if an INIT_REQ message has not been received. When this variable equals 0 no transmissions will occur.
_P1_RBE_STATE	%MB 3.8432 SINT	4.60	Indicates the current state of this RBE port. See the 'RBE' function block online help to learn the meaning of the number.
_P1_MAX_SLAVES	%MD 3.8440 USINT	4.50	Maximum number of slaves allowed below a virtual node.
_P1_TOP_LEVEL_NODES	%MD 3.8441 USINT	4.50	The number of slaves under a master (or virtual nodes under an Emaster.)
_P1_TOTAL_NODES	%MD 3.8442 : INT	4.50	Total number of slaves on this port.
_P1_DEAD_ARRAY	%MD 3.8444 : INT	4.50	This array is similar to the _SLAVE_DEAD array. It is applicable to the EBSAP Master only. <i>_Pn_DEAD_ARRAY</i> is the number of a registered BOOL array that represents the number of slave nodes on Port <i>n</i> . The array must be dimensioned by the number of virtual nodes and <i>max_slaves</i> for row and column, respectively. The array elements are used by the system to report the current status of the slaves. By default, when a node is not responding to the POLLS it is declared as dead and the corresponding BOOL status is set to TRUE. NOTE: This default logic for interpreting the BOOL can be reversed via <i>_BSAP_FLAG_SENSE</i> .
_P1_DISABLE_ARRAY	%MD 3.8446 : INT	4.50	This array is similar to the <i>_SLAVE_POLL_DIS</i> array. It is applicable to the EBSAP Master only. <i>_Pn_DISABLE_ARRAY</i> is the number of a

Port 1 – offsets 8384-8428			
			registered BOOL array that represents the number of slave nodes on Port <i>n</i> . The array must be dimensioned by the number of virtual nodes and max_slaves for row and column, respectively. The array elements are used as an input to the communication system to indicate whether the slave(s) are to be polled or not. By default, when an array element is set to TRUE the corresponding node is not polled. NOTE: This default logic for interpreting the BOOL can be reversed via <code>_BSAP_FLAG_SENSE</code> .
<code>_P1_PAD_FRONT</code>	<code>%MD 3.8448 USINT</code>	4.60	Number of front pad characters for BSAP messages.
<code>_P1_PAD_BACK</code>	<code>%MD 3.8449 USINT</code>	4.60	Number of back pad characters for BSAP messages.
<code>_P1_STATISTICS_ARRAY</code>	<code>%MD 3.8450 INT</code>	5.30	This designates an array to maintain statistics for an I/O Expansion Rack using RS 485 communication. For details on the array structure and statistics, see the online help in ControlWave Designer.
Port 2 – offsets 8512-8556 (replication of port 1 offsets as applicable).			
Port 3 – offsets 8640-8684 (replication of port 1 offsets as applicable).			
Port 4 – offsets 8768-8812 (replication of port 1 offsets as applicable).			
Port 5 – offsets 8896-8940 (replication of port 1 offsets as applicable).			
Port 6 – offsets 9024-9068 (replication of port 1 offsets as applicable).			
Port 7 – offsets 9152-9196 (replication of port 1 offsets as applicable).			
Port 8 – offsets 9280-9324 (replication of port 1 offsets as applicable).			
Port 9 – offsets 9408-9452 (replication of port 1 offsets as applicable).			
Port 10 – offsets 9536-9580 (replication of port 1 offsets as applicable).			
Port 11 – offsets 9664-9708 (replication of port 1 offsets as applicable).			
Port 12 – offsets 9792-9836 (replication of port 1 offsets as applicable).			
<code>_S9_IO_BOARD_ID_STR</code>	<code>%MB 3.11000 :STRING</code>		ID string for 9 th I/O board.
<code>_S10_IO_BOARD_ID_STR</code>	<code>%MB 3.11085 :STRING</code>		ID string for 10 th I/O board.

Port 1 – offsets 8384-8428			
_S11_IO_BOARD_ID_STR	%MB 3.11170	:STRING	ID string for 11 th I/O board.
_S12_IO_BOARD_ID_STR	%MB 3.11255	:STRING	ID string for 12 th I/O board.
_S13_IO_BOARD_ID_STR	%MB 3.11340	:STRING	ID string for 13 th I/O board.
_S14_IO_BOARD_ID_STR	%MB 3.11425	:STRING	ID string for 14 th I/O board.

System Variable Name	Address	Data Type	Minimum Firmware Needed	Description
_IPn_RBE_ERE_COUNT	%MD 3.12000	UDINT	4.50	Count of the number of Exception Reports (ER) generated and posted to the Report Pool for Ethernet port <i>n</i> . This count is incremented each time a new exception is reported. Application may reset this count.
_IPn_RBE_RM_COUNT	%MD 3.12004	UDINT	4.50	Count of the number of Report Messages successfully transmitted for Ethernet port <i>n</i> . Each RM can have more than 1 Exception Report in it. This count is not incremented when RM are retransmitted. Application may reset this count.
_IPn_RBE_POOL_OVERFLOW	%MD 3.12008	UDINT	4.50	Count of the number of Exception Reports the RBE FB wanted to post in the Report Pool but could not because the Report Pool was full. These exceptions may essentially be lost for the RBE Client at this port. Application may reset this count.
_IPn_RBE_REXMIT_COUNT	%MD 3.12014	USINT	4.50	Indicates number of retransmissions of the last Report Message. This happens when an expected RBE_ACK is not received from the RBE Client. Following the RBE_ACK_TMO the last RM is retransmitted. On such retransmissions the RM_COUNT is not incremented. This count is automatically reset back to 0 whenever an ACK is received.
_IPn_RBE_PENDING	%MX 3.12019.1	:BOOL	4.50	This variable is set to TRUE when there is one or more Exception Reports pending for Ethernet port <i>n</i> . It will automatically be changed to FALSE when the last Exception Report from the Report Pool has been successfully transmitted.
_IPn_RBE_RM_SINCE_ACK	%MD 3.12015	USINT	4.50	Indicates number of Report Messages sent since last ACK was received at Ethernet Port <i>n</i> . Count is adjusted whenever an ACK is received. This count is continuously incremented for each Report Message sent when the ACK Limit is set to 0.

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
_IPn_IP_RBE_ADDRESS	%MD 3.12128 UDINT	4.50	Displays the IP Address for Ethernet Port <i>n</i> .
_IPn_RBE_POOL_SIZE	%MW 3.12012 UINT	4.50	The capacity of the Report Pool for port Ethernet Port <i>n</i> . This size governs the maximum number of Exception Reports that can be outstanding at any given time without being successfully reported. This size is determined based on the expected rate of exception occurrence and the throughput at the port. This size can be changed on line. However, for the new size to be effective the RBE FB must be re-run with the ioablnit parameter in TRUE state. When this variable is set to 0 (default), the RBE is not active for the port. Adjust this parameter only if the Report Pool overflows or if the system wide RAM usage needs to be redistributed.
_IPn_RBE_ERE_FORMAT	%MD 3.12017 USINT	4.50	Exception Report format type for Ethernet port <i>n</i> .
_IPn_RBE_ACK_LIMIT	%MD 3.12018 USINT	4.50	RBE Acknowledge Limit for Ethernet Port <i>n</i> . Indicates if and when an ACK is expected from the RBE Client. 0 = Never wait for an ACK for Report Message (default) n = 1-127 = send up to n Report Messages (RM) then stop if an ACK is not received for any of the Report Messages. When the reporting has stopped while waiting for an ACK, a timeout will result in the retransmission of the last RM. The periodic retransmission will occur until an ACK is received.
_IPn_RBE_REPEAT_TIMEOUT	%MD 3.12020 DINT	4.50	Specifies the time to wait for an RBE ACK at Ethernet Port <i>n</i> . If an ACK is not received within this period then retransmit the last Report Message. Periodically repeat this until an RBE ACK is received from the client. Ignore unrelated or out of sequence ACKs. A new Init or Demand Request will force an exit from this state. This timeout is also used during the initialization stage to determine when a WAITING INIT message will be re-sent if an INIT_REQ message has not been received. When this variable equals 0 no transmissions will occur.
_IPn_RBE_STOP_RPT_MSG	%MX 3.12019.0 : BOOL	4.50	Control report message transmissions for port <i>n</i> . FALSE = Start/Restart Report Message transmission at port <i>n</i> . TRUE = Temporarily

System Variable Name	Address Data Type	Minimum Firmware Needed	Description
			stop sending Report Message for port <i>n</i> . When restarting from the stopped state the RM will begin with the next new Exception Report. This does not prevent new Exception Reports from being put into the buffer pool. It is possible to overflow the buffer pool while this variable is set.
_IPn_RBE_USE_ACCOL_NAME	%MX 3.12019.2 : BOOL	4.50	This variable determines the format of the variable name in the Exception Report. TRUE = the variable names are in ACCOLII format. FALSE= the variable names are in IEC1131 format.
_IPn_RBE_GO_ACT_ON_START	%MX 3.12019.3 : BOOL	4.50	Variable determines the initial state for RBE at Ethernet Port <i>n</i> following an application COLD or WARM start. FALSE (default) = RBE is to send the WAITING_INIT message to the RBE Client and wait for the INIT_REQ message. Periodically, ACK_TMO, repeat the wait message until the init message is received. TRUE = Following a successful initialization the RBE enters the active state where the database scan and exception reporting take place.
_IPn_RBE_STATE	%MB 3.12024	4.60	The current state of RBE for the Ethernet port. See the 'RBE function block' online help to learn the meaning of the number.

Static Memory Area:

In addition to the system variables described, a static memory array exists beginning at offset 3.100000. The static memory area is useful for storing data you have accumulated, and don't want to lose if the controller is restarted, such as flow totals, equipment run times, etc. This memory is normally NOT written to by the system, and so can be used to save user information across restarts. This defaults to 16K but can be reduced if there is insufficient space in SRAM for variables marked 'RETAIN'. This memory will only be re-initialized on a re-start if the SRAM control switch is set OFF, or if there is no battery backup.

It can also be initialized under program control by setting its value to **0**.

Example - Manually Defining a System Variable to Display the Task Priority

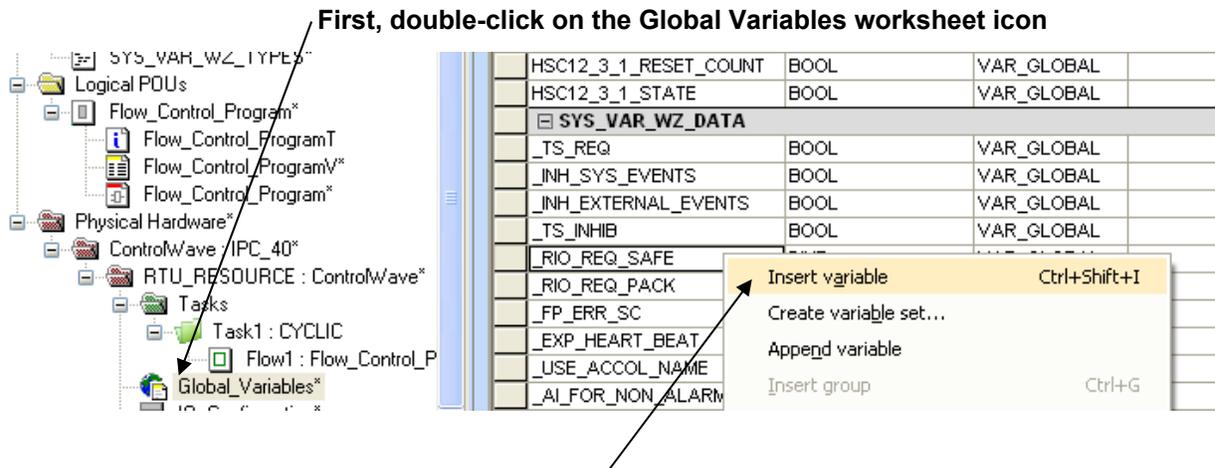
Let's say that within one of our programs we want to know the configured priority of the first task in the project:

To find out if this information is available, look through the System Variable Mapping Charts and you will find the entry:

_T1_TASK_PRIO	%MW 1.1014 : INT	The configured priority of this task.
---------------	------------------	---------------------------------------

Our _T1_TASK_PRIO variable would be defined in a variable worksheet. For this particular project, it's called **RTU_RESOURCEV** but depending upon which version of ControlWave Designer you are using, it could be named 'Global Variables' or something else.

- First, double-click on the 'Global Variables' icon in the project tree, to bring up the worksheet.
- Then right-click in the 'Global Variables' section, and choose "Insert variable" from the pop-up menu.



Next, right-click within the worksheet, and choose "Insert variable" from the pop-up menu.

- A new variable called 'NewVar1' will appear in the worksheet. Double-click on the 'NewVar1' name to edit it. Enter the name '_T1_TASK_PRIO' from the table.

Double-click on the "NewVar1" name to edit it.

SYS_VAR_WZ_DATA			
_TS_REQ	BOOL	VAR_GLOBAL	
_INH_SYS_EVENTS	BOOL	VAR_GLOBAL	
_INH_EXTERNAL_EVENTS	BOOL	VAR_GLOBAL	
_TS_INHIB	BOOL	VAR_GLOBAL	
NewVar1	BOOL	VAR_GLOBAL	
_RIO_REQ_SAFE	DINT	VAR_GLOBAL	

Click in the "Type" field to change the Type.

- Now click on the 'Type' field, and a list box will appear. Choose 'INT' because that is the type specified in the table.

When you have finished editing the "Name" and "Type" they should look like this.

_TS_INHIB	BOOL	VAR_GLOBAL	
_T1_TASK_PRIO	INT	VAR_GLOBAL	
_RIO_REQ_SAFE	DINT	VAR_GLOBAL	
_RIO_REQ_PACK	DINT	VAR_GLOBAL	

- Drag the scroll bar to bring additional fields into view.
- Now, enter the address for the variable, as shown in the table '%MW 1.1014'. When this is done, and the worksheet is closed, you have successfully defined a global variable for the task priority.

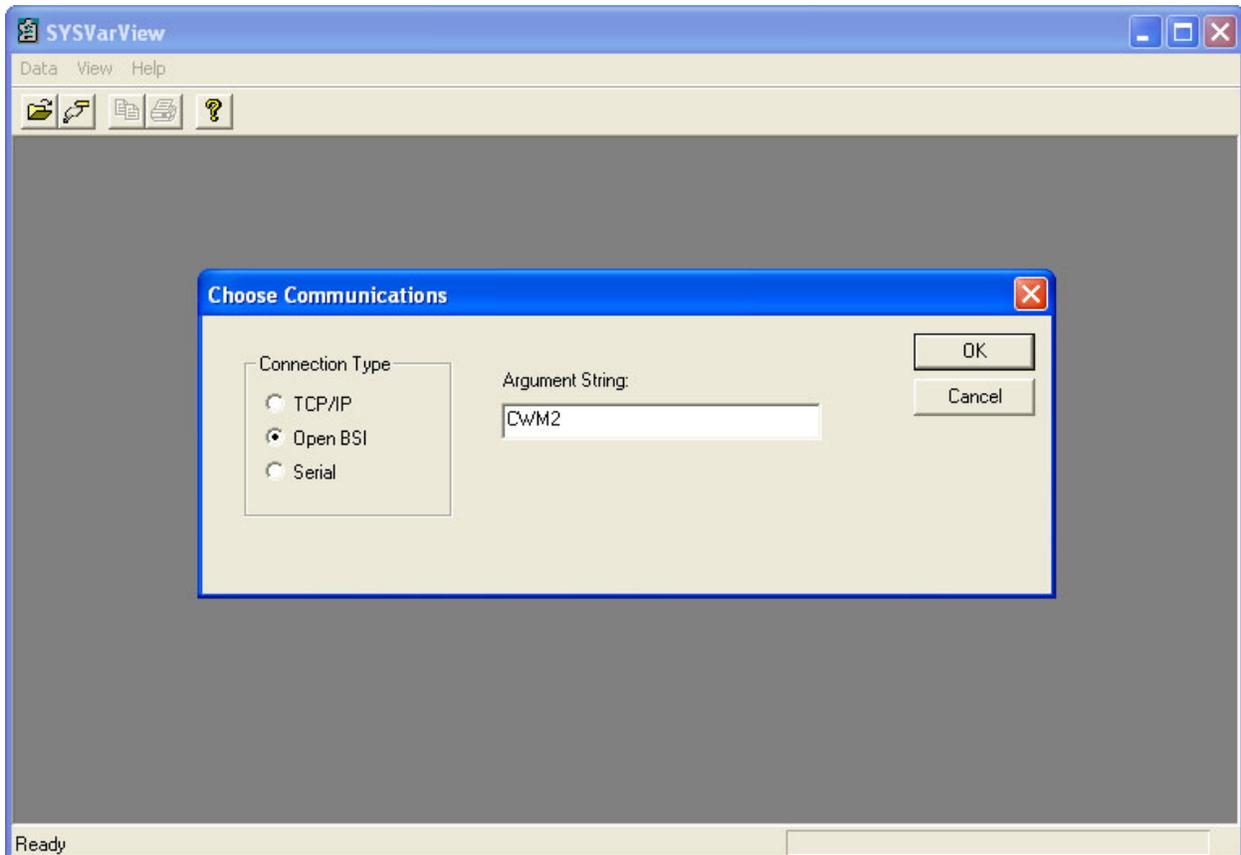
Enter the address here

_TS_INHIB	%MX 3.0.3	FALSE
_T1_TASK_PRIO	%MW 1.1014	
_RIO_REQ_SAFE	%MD 3.28	50

Using the System Variable Viewer

OpenBSI includes a debugging tool for viewing the current value of system variables in the ControlWave.

To start the System Variable Viewer, click **Start** → **Programs** → **OpenBSI Tools** → **Debugging Tools** → **System Variable Viewer**.



In the System Variable Viewer, click **Data** → **Communications** to call up the Choose Communications dialog box.

There are three possible connection types:

TCP/IP: Specify the IP address, and timeout (in milliseconds).

Syntax is: **-ip** *ip_address* *timeout*

Example **Argument String:** -ip 10.211.74.222 2000

Open BSI: Specify the node name of the ControlWave to which you want to communicate for the **Argument String**.

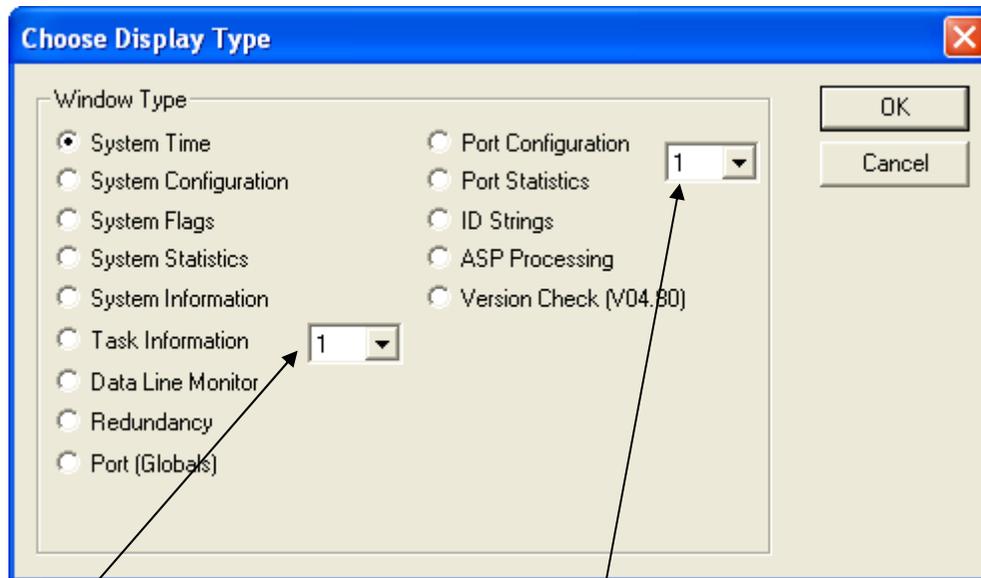
Example: CWM2

Serial: Specify the PC communication port, baud rate, and timeout (in milliseconds).

Syntax is: *comm._port baud_rate timeout*

Example **Argument String:** COM1 9600 2000

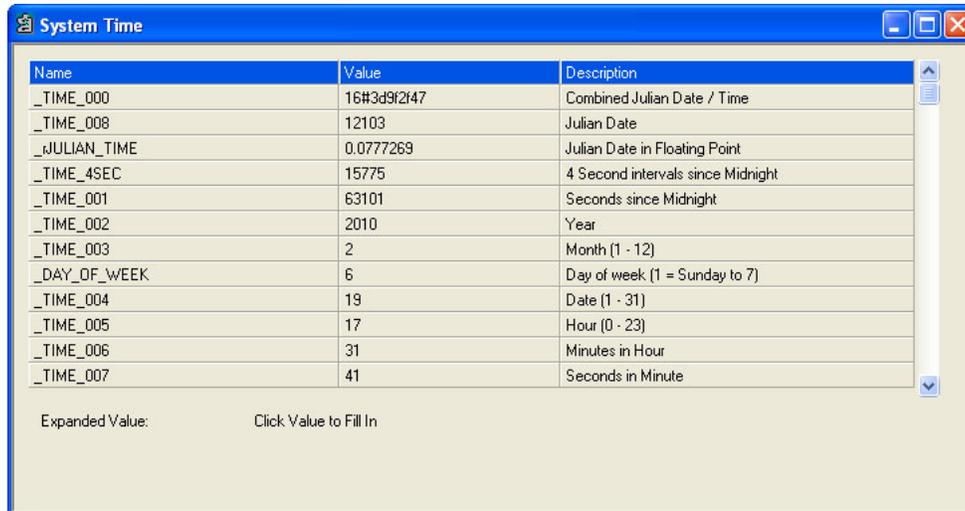
Click **OK** when you complete the argument string. This calls up the Choose Display Type dialog box.



If you want to view variables for a task other than Task 1, choose it here.

If you want to view variables for a port other than Port 1, choose it here.

In the Choose Display Type dialog box. Specify the list of system variables you want to view. If you want to see Task Information or Port information, you'll need to choose the proper numbered task or port first. After you make your selection, click **OK** and the list opens.



The screenshot shows a window titled "System Time" with a table of system variables. The table has three columns: Name, Value, and Description. Below the table, there is an "Expanded Value" field and a "Click Value to Fill In" button.

Name	Value	Description
_TIME_000	16#3d9f2147	Combined Julian Date / Time
_TIME_008	12103	Julian Date
_JULIAN_TIME	0.0777269	Julian Date in Floating Point
_TIME_4SEC	15775	4 Second intervals since Midnight
_TIME_001	63101	Seconds since Midnight
_TIME_002	2010	Year
_TIME_003	2	Month (1 - 12)
_DAY_OF_WEEK	6	Day of week (1 = Sunday to 7)
_TIME_004	19	Date (1 - 31)
_TIME_005	17	Hour (0 - 23)
_TIME_006	31	Minutes in Hour
_TIME_007	41	Seconds in Minute

Expanded Value: Click Value to Fill In

Notes:

- The tool prompts you to log in to the RTU to view the system variables.
 - If you click on a particular value, it appears in the **Expanded Value** field.
 - Refreshing of the screen may be slow.
 - To choose a different list to view, click **File → New List**.
-

Variable Extension Wizard

What is the Variable Extension Wizard?

The **Variable Extension Wizard** is run from within ControlWave Designer (this feature was added with OpenBSI Version 5.4/ControlWave Designer Version 4.0). It allows you to create initialization files (*.INI) which assist in batch configuration of variables within the ControlWave-series controller.

The information in these initialization files is incorporated into the ControlWave project when read using either the DB_LOAD or RBE function blocks. The initialization files may be used to:

- Configure lists (Requires ControlWave firmware 04.40 or newer)
- Identify variables which should be collected via Report by Exception (Requires ControlWave firmware 04.40 or newer)
- Configure alarms (Requires ControlWave firmware 04.90 or newer)
- Configure descriptive text, ON/OFF text, inhibit/enable flags, or units text (Requires ControlWave firmware 04.90 or newer)

Important

The configuration information entered via this method is **not** visible within ControlWave Designer. For example, you will **not** see LIST function blocks for lists created via this method, since it is performed via initialization files.

Before You Begin

- In order to view variables in the Variable Extension Wizard, you must have marked those variables for PDD collection within the project *before* you run the Variable Extension Wizard.
- Your ControlWave project must be in a state where it can be compiled successfully.

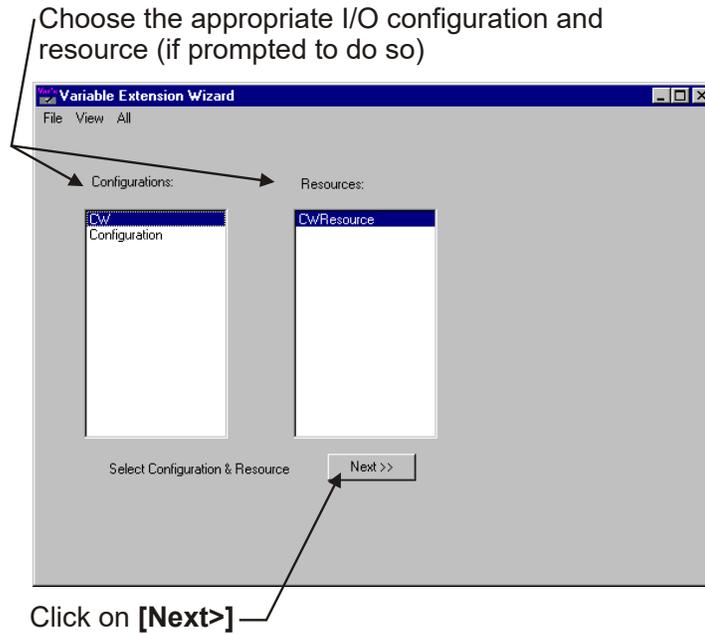
Starting the Variable Extension Wizard

The Variable Extension Wizard is run from within ControlWave Designer. To do this, click on:

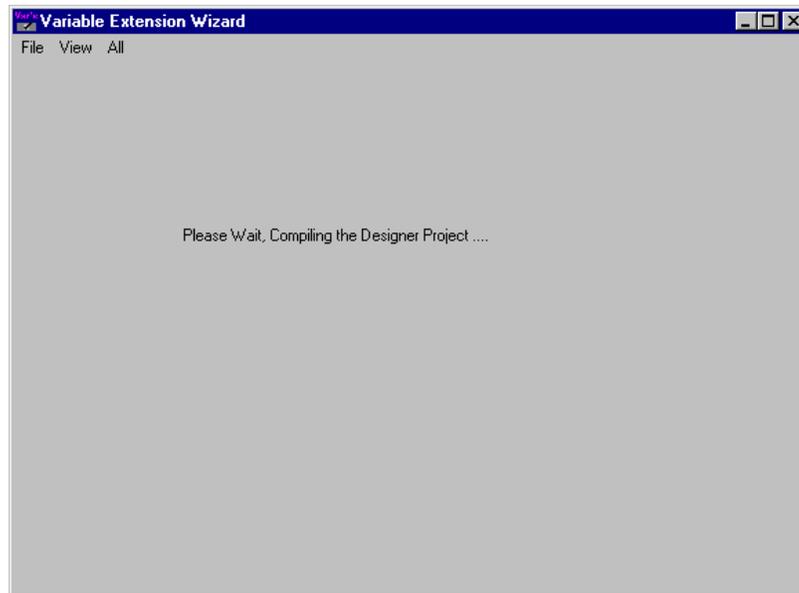
View → Variable Extension Wizard

Using the Variable Extension Wizard

If your project includes more than one resource, you will be prompted to choose which resource and I/O Configuration you want to work with. Choose the resource, then click **[Next>]**.



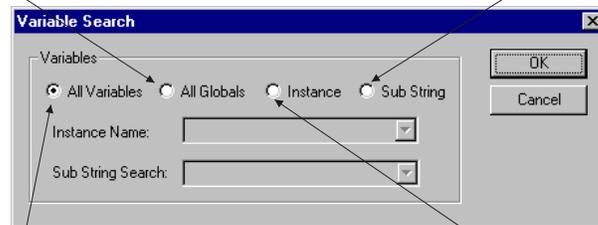
When the Variable Extension Wizard first starts it will compile your ControlWave project.



When the compilation has completed successfully, the Variable Search dialog box will appear.

Search only for variables with a given string in the variable name. (You must enter that string in the **“Sub String Search”** field.)

Search only for global variables.



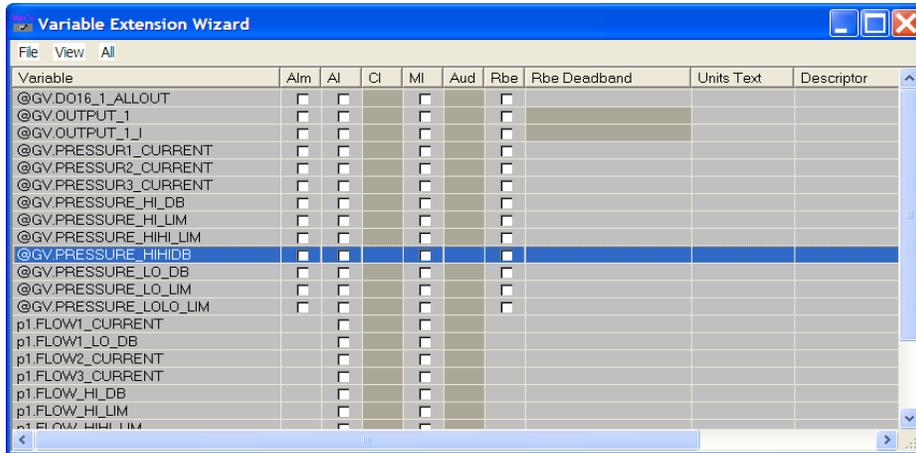
Search for all global variables and all variables marked 'PDD'.

Search only for variables with a given instance name. (You must enter that in **“Instance Name”**.)

In the Variable Search dialog box, choose the type of variables you want to search for.

- | | |
|----------------------|---|
| All Variables | Selecting this will cause the Wizard to search for all variables marked PDD, as well as all global variables (those with an instance name of @GV). |
| All Globals | Selecting this will cause the Wizard to search for all global variables in the project (those with an instance name of @GV). |
| Instance | Selecting this will cause the Wizard to search only for variables from a particular program instance. You must enter the name of that program instance in the “Instance Name” field. |
| Sub String | Selecting this will cause the Wizard to search only for variables whose names include a particular string of text. You must enter that text string in the “Sub String Search” field. |

When you've chosen which variables to search for, click **[OK]** and the variables display in a Search Grid.



Drag the scroll bar to bring more variables

Once the variables are displayed in the Search Grid you can do any of the following:

- Mark the variable for Report By Exception (RBE) collection, and, if an analog variable, configure its deadband value.
- Configure the variable as an alarm.
- Create one or more lists containing variables.
- Assign units (for analogs) or ON/OFF text (for BOOLS) to the variable.
- Set initial values of manual or alarm inhibit/enable flags.
- Create descriptive text for the variable.

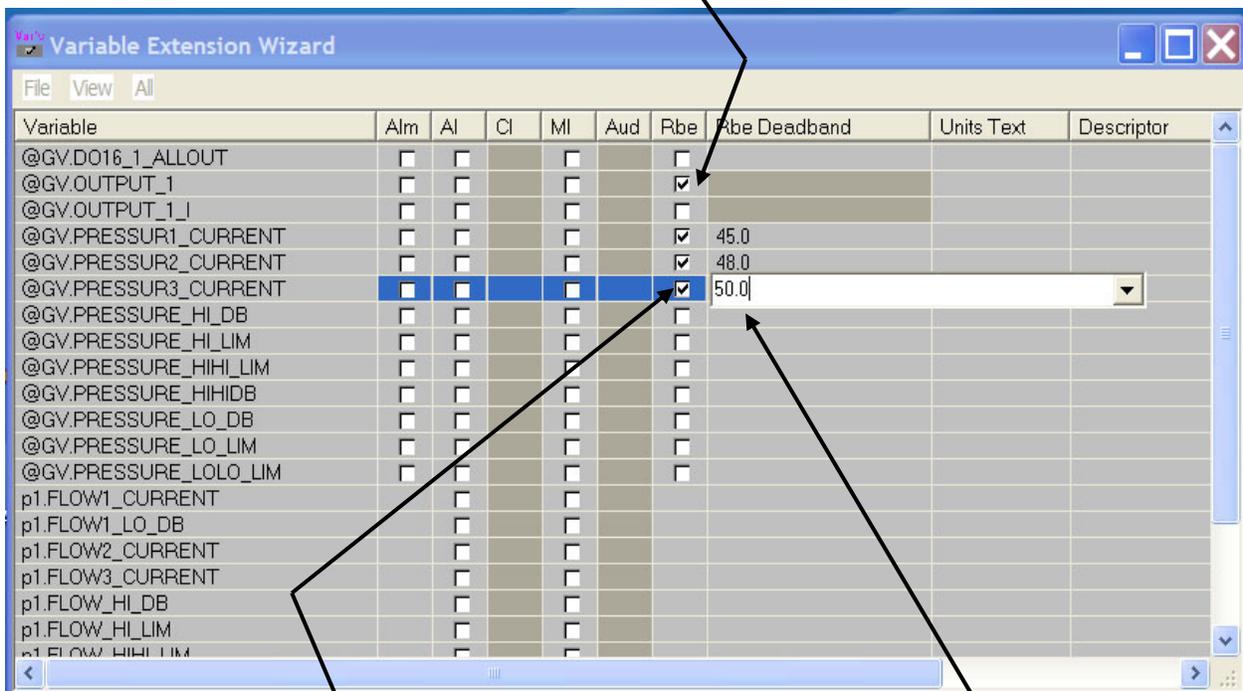
Marking a Variable for Report by Exception (RBE) Collection

RBE is a method of data collection where variables are collected **only** when they change.

For logical variables (BOOL), this means whenever the variable goes from TRUE to FALSE or FALSE to TRUE.

For analog variables (INT, REAL, etc.), this means whenever the analog variable changes more than a pre-configured deadband value. You **MUST** configure a deadband for an analog RBE variable, or else any change at all, even a minor fluctuation, would cause the variable to be collected.

To select a BOOL variable for RBE collection, just select its “Rbe” checkbox.



To mark an analog variable for RBE collection, first select its “Rbe” checkbox, then enter a deadband in the “Rbe Deadband” field, or use the list box for the field to select a variable which holds a value that will serve as the deadband.

NOTE: You must press the **[Enter]** key on the keyboard to complete the entry, or else you cannot exit the field.

Note:

Your project must include an RBE function block, and you must have configured it, in order for these variables to be collected via RBE.

For a logical RBE variable

To define a logical RBE variable, simply check the “**Rbe**” check box for that variable.

For an analog RBE variable

To define an analog RBE variable, check the “**Rbe**” check box for that variable, then enter a deadband value in the “**Rbe Deadband**” field, or use the list box for the field to select *another* variable whose value will be used as the RBE deadband. **Note:** You MUST press the [Enter] key on your keyboard to exit the “**Rbe Deadband**” field.

Marking / Unmarking all variables in the Search Window for RBE collection

To mark all variables in the Search Grid for RBE collection, click on **All → Mark All RBE**. You will still have to define RBE deadbands for all analog RBE variables.

To un-mark all variables in the search grid, so that none of them will be collected by RBE, click on: **All → Clear All RBE**

Configuring a Variable as an Alarm

The controller generates alarm messages in response to a significant change in a variable's value or status. Full details on how alarms work are explained in the *Alarm Configuration* section in this manual. Alarms are configured in one of two ways:

1. Using Alarm function blocks (See the *Alarm Configuration* section, earlier in this manual)
2. Using the Variable Extension Wizard

Configuring an Analog Alarm Variable

Click on the “**Alm**” check box for the variable. The Analog Alarms dialog box will appear.

Specify alarm limits for the variable. These limits determine at what point the variable enters an alarm condition. The alarm limit values may be entered directly as a constant, or you may specify separate analog variables to hold the value of each alarm limit.

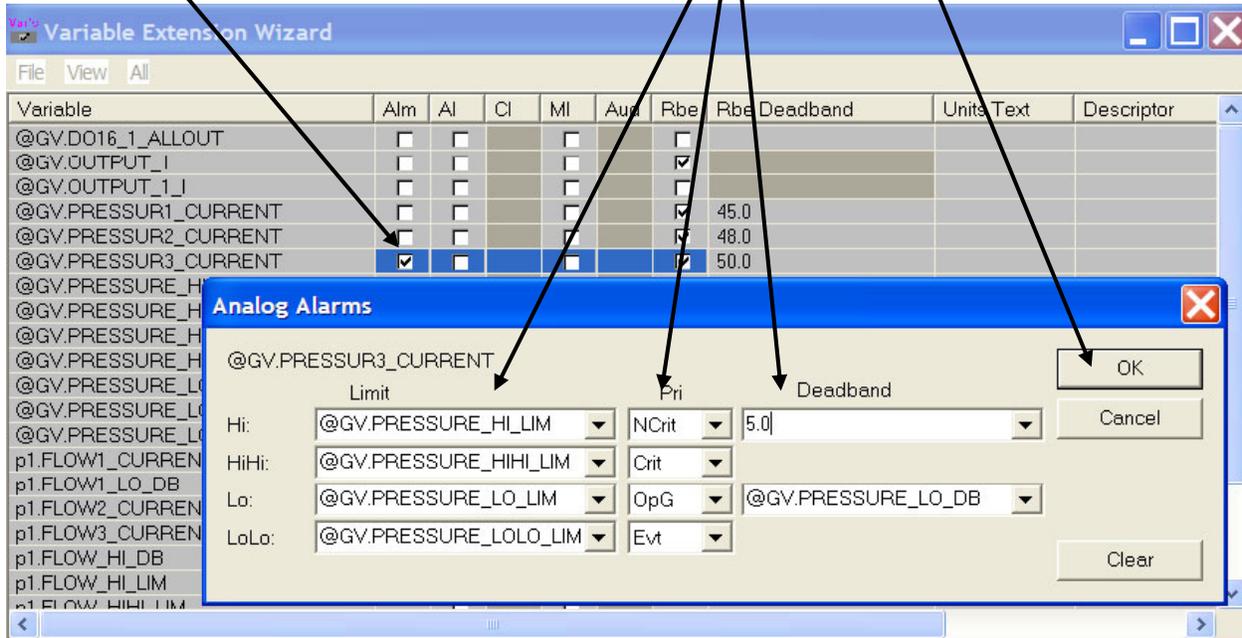
Specify a high deadband value, and/or low deadband value for the variable. Deadbands are used to define a range around the alarm limits where a minor fluctuation of the variable should not change the ‘in alarm’ or ‘normal state’ condition of the variable. Again, it may be entered as a constant, or as a separate analog variable.

Choose priorities for each alarm limit. These are used to specify the severity of the alarm condition. The priorities from least important to most important are: Event ‘Evt’, Operator Guide ‘OpG’, Non-Critical ‘NCrit’, and Critical ‘Crit’.

Click **[OK]** when finished.

First, select the **"Alm"** box for the variable.

Then specify alarm limits, priorities, and deadbands. When you're done click on **[OK]**.



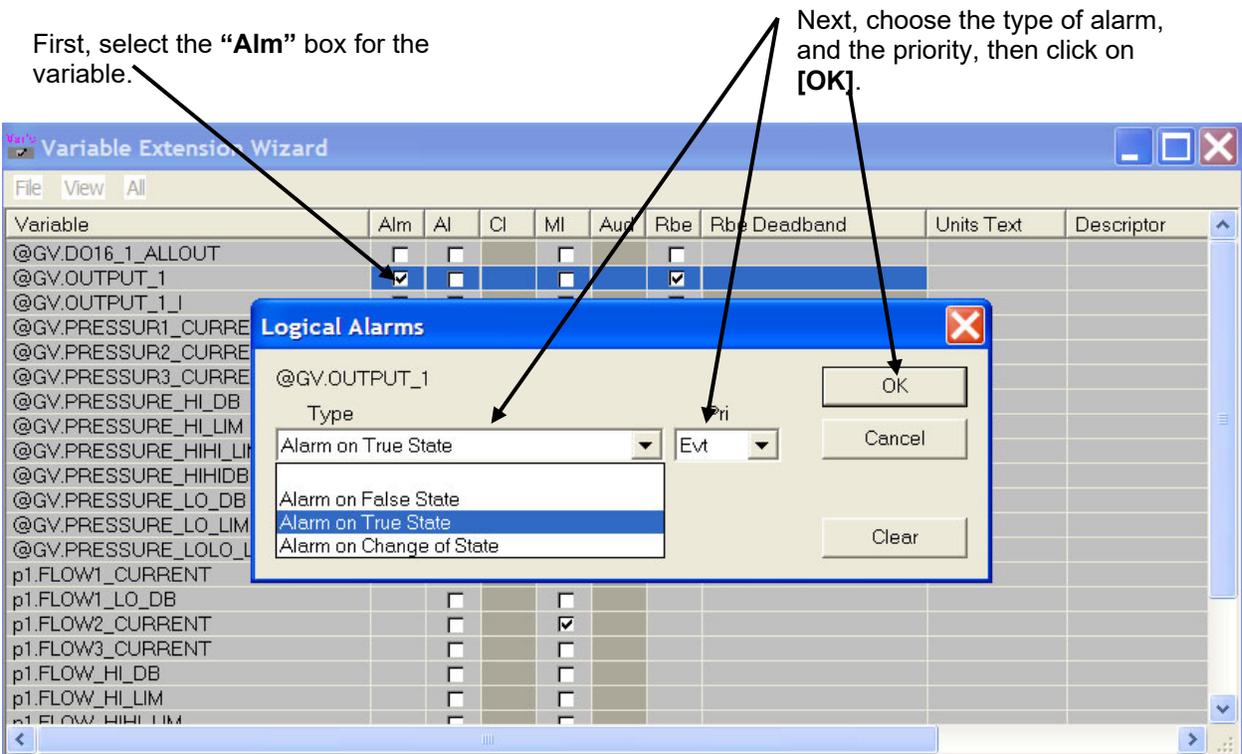
Configuring a Logical (BOOL) Alarm Variable

Click on the **“Alm”** check box for the variable. The Logical Alarms dialog box will appear.

Choose the **“Type”** of the alarm: ‘Alarm on True State’ means that an alarm is generated when the variable changes from FALSE to TRUE; ‘Alarm on False State’ means that an alarm is generated when the variable changes from TRUE to FALSE; ‘Alarm on Change of State’ means that an alarm is generated by any change.

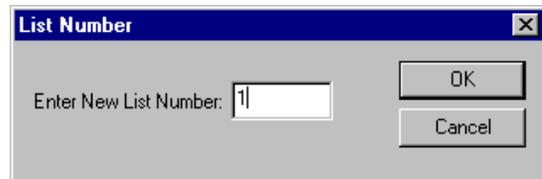
Choose a priority for the alarm, to indicate its severity. The priorities from least important to most important are: Event ‘Evt’, Operator Guide ‘OpG’, Non-Critical ‘NCrit’, and Critical ‘Crit’.

Click on **[OK]** when finished.



Creating / Editing a List

To create a list of variables, click on **View → Lists**.



The Lists dialog box will appear. If you want to edit an existing list, select it from the **“List Number”** list box. To create an all new list, click **[New]** and enter a number for the list.

To add a variable to the current list, click on its name in the “Searched Variable Directory” then click on the [>>] button. To remove a variable from the current list, click on its name in the “List Entries” field then click on the [<<] button.

You can optionally enter a description for the list in the “List Descriptor” field.

If you want users to be able to edit the list, online, check the “Allow on-line edits” box.

Note: Users can delete variables; but can only add variables that already exist in the project.

To delete an existing list, choose the list number, then click [Delete]. **Note:** There is no prompt to confirm the deletion – the entire list is deleted immediately from the wizard. (OpenBSI 5.7 Service Pack 2 and newer.)

Note:

Your ControlWave project must include a configured DB_LOAD function block to load these lists.

Options:

- Optional, you can enter a description for the list here.
- If you want users to be able to make online edits to signal lists, check this box.
- Click on “New” to create a new empty list.
- Variables are stored and referenced in the list in the order in they appear under “List Entries.” To change the position of a variable, click on it in “List Entries” and then click on the [Move Up] or [Move Down] buttons to position it in the desired order.

Actions:

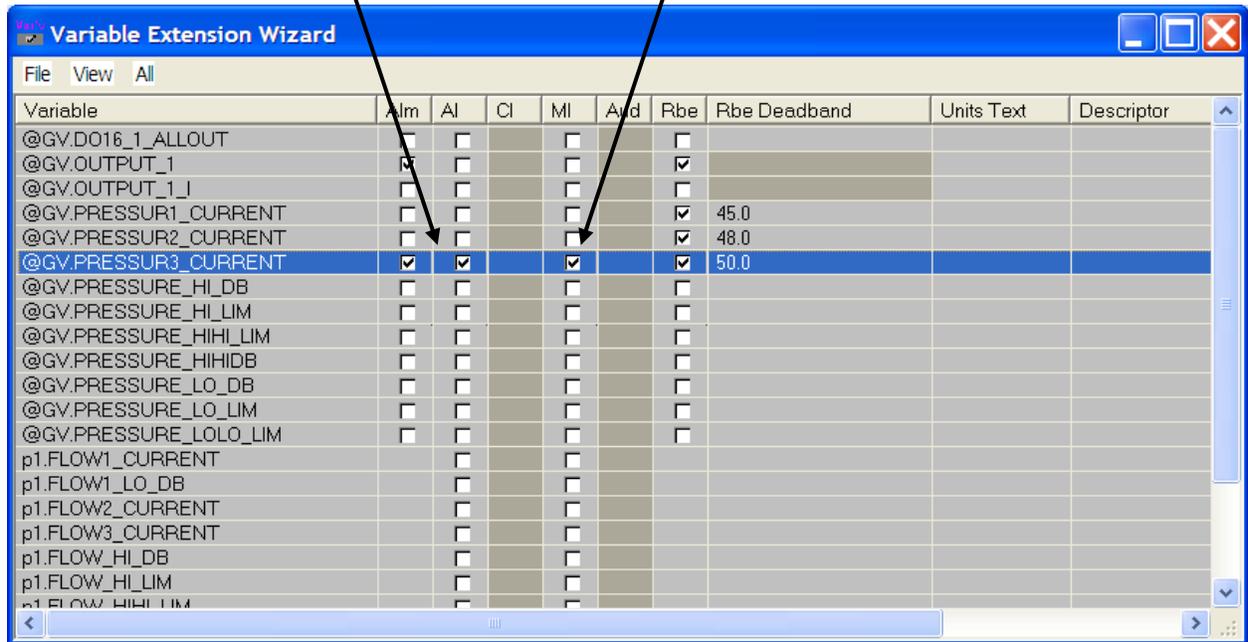
- To edit a pre-existing list, select it from the list box.
- To add a variable to the current list, click on the variable name, then click on the [>>] button to add it to the “List Entries.”
- To remove a variable to the current list, click on the variable name in “List Entries,” then click on the [<<] button.

Setting initial values for Manual or Alarm Inhibit/Enable Flags

To set the initial value of a Manual Inhibit/Enable flag or Alarm Inhibit/Enable flag to Inhibit, simply select the “MI” or “AI” checkboxes, respectively.

To set a variable's initial Alarm Inhibit/Enable status to 'Inhibit', check the **AI** box.

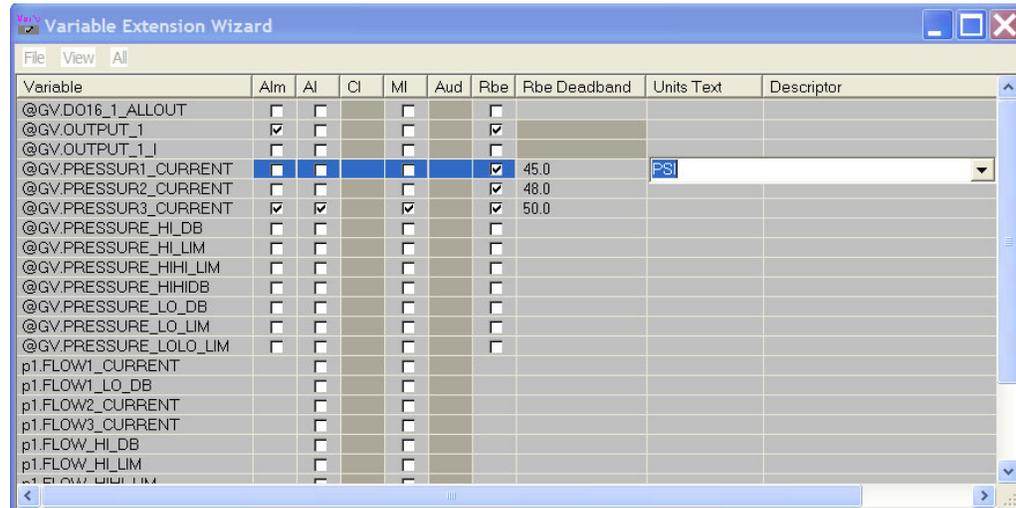
To set a variable's initial Manual Inhibit/Enable status to 'Inhibit', check the **MI** box.



Assigning Units Text (Analog Variables ONLY)

To assign engineering units to a variable, click in the “Units Text” field, and enter up to six characters of units text, then press the [Enter] key on the keyboard, to exit the field.

Alternatively, you can specify a variable name in the field, which will hold the units text.



Note: You *can* enter units text in brackets in the **Description** field of a variables worksheet in ControlWave Designer.

Name	Description
System_Variables	
WATER_TEMP	[DEGC]

The units text entered there, however, only appears within ControlWave Designer; it is not downloaded to the RTU. It cannot be viewed in DataView or other data collection programs.

You can view this text in the Variable Extension Wizard, however, it appears in green to indicate that it does not become part of the _VARDEFS.INI file and is not downloaded to the RTU.

If you want to convert this text so it *can* be downloaded to the RTU, position your cursor in the units text field of the Variable Extension Wizard, and press the [Enter] key.

Position the cursor in the field.



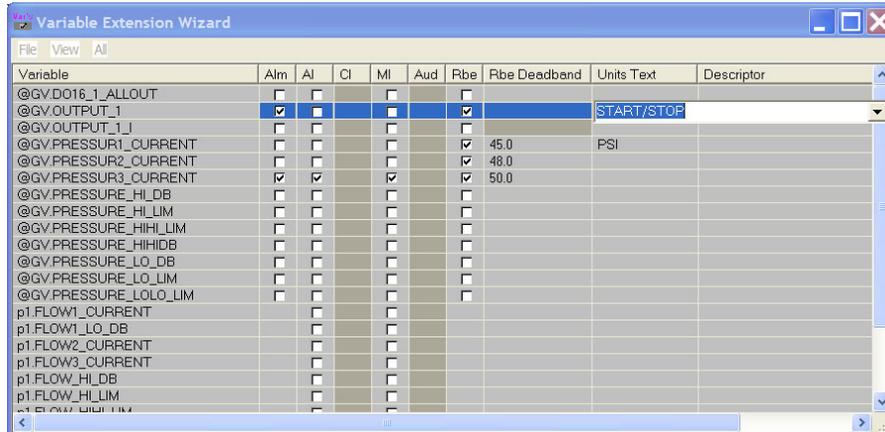
The text now changes color from green to black, and can be downloaded as if it was entered directly in the Variable Extension Wizard.



Assigning ON/OFF Text (BOOL Variables ONLY)

To assign ON/OFF text to a BOOL variable, click in the “Units Text” field, and enter up to six characters of ON text, followed by a slash “/”, and then up to six characters of OFF text. Press the [Enter] key on the keyboard, to exit the field.

Alternatively, you can specify a variable name in the field, which will hold the ON/OFF text.



You *can* enter ON/OFF text in brackets in the Description field of a variables worksheet in ControlWave Designer.

Name	Type	Usage	Description
System_Variables			
SECURITY_DOOR	BOOL	VAR_GLOBAL	[OPENED/CLOSED]

The ON/OFF text entered there, however, only appears within ControlWave Designer; it is not downloaded to the RTU. It cannot be viewed in DataView or other data collection programs.

You can view this text in the Variable Extension Wizard, however, it appears in green to indicate that it does not become part of the _VARDEFS.INI file and is not downloaded to the RTU.

If you want to convert this text so it *can* be downloaded to the RTU, position your cursor in the units text field of the Variable Extension Wizard, and press the [Enter] key on the keyboard.

Position the cursor in the field.



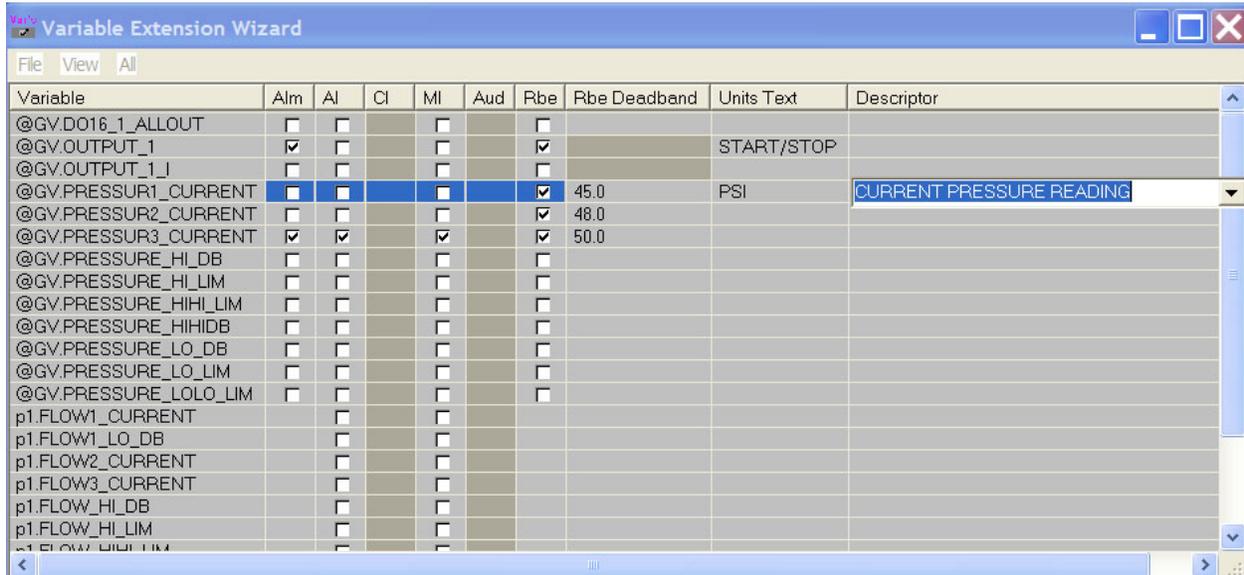
The text now changes color from green to black, and can be downloaded as if it was entered directly in the Variable Extension Wizard.

@GV.SECURITY_DOOR									OPENED/CLOSED	
-------------------	--	--	--	--	--	--	--	--	---------------	--

Creating Descriptive Text for the Variable

To create descriptive text for a variable, click in the “**Descriptor**” field, and enter up to 64 characters of descriptive text, then press the [Enter] key on the keyboard, to exit the field.

Alternatively, you can specify a variable name in the field, which will hold the descriptive text.



You *can* enter descriptive text in the **Description** field of a variables worksheet in ControlWave Designer.

Name	Type	Usage	Description
TEST_IN_PROGRESS	BOOL	VAR_GLOBAL	
START_TEST	BOOL	VAR_GLOBAL	PROCESS TEST HAS STARTED

The descriptive text entered there, however, only appears within ControlWave Designer; it is not downloaded to the RTU. It cannot be viewed in DataView or other data collection programs.

You can view this text in the Variable Extension Wizard, however, it appears in green to indicate that it does not become part of the _VARDEFS.INI file and is not downloaded to the RTU.

If you want to convert this text so it *can* be downloaded to the RTU, you have two options:

Option 1: Identify the text for each individual variable:

If you only want the descriptive text for certain variables to be downloaded to the RTU you must position your cursor in the descriptor field of the Variable Extension Wizard, and press the [Enter] key.

Position the cursor in the field.



The screenshot shows a table with columns: Variable, Alm, AI, CI, MI, Aud, Rbe, Rbe Deadband, Units Text, and Descriptor. The first row contains the variable '@GV.START_TEST'. The 'Descriptor' column for this variable contains the text 'PROCESS TEST HAS STARTED'. A black arrow points from the text 'Position the cursor in the field.' to the 'PROCESS TEST HAS STARTED' text in the Descriptor column.

Variable	Alm	AI	CI	MI	Aud	Rbe	Rbe Deadband	Units Text	Descriptor
@GV.START_TEST	<input type="checkbox"/>			PROCESS TEST HAS STARTED					

The text now changes color from green to black, and can be downloaded as if it was entered directly in the Variable Extension Wizard. Repeat this process for each variable you want to have descriptive text.



The screenshot shows the same table as above, but the text 'PROCESS TEST HAS STARTED' in the Descriptor column is now black, indicating it has been processed.

Variable	Alm	AI	CI	MI	Aud	Rbe	Rbe Deadband	Units Text	Descriptor
@GV.START_TEST	<input type="checkbox"/>			PROCESS TEST HAS STARTED					

Option 2: Specify that you want descriptive text for ALL variables to be downloaded

If you want the descriptive text entered in variable worksheets for ALL variables to be downloaded to the RTU, click **All > Store All Descriptors**. **Note:** This method requires OpenBSI 5.9 or newer.

Saving the Initialization Files and Exiting the Wizard

To save the initialization files, click on **File → Save**.

To exit the wizard, click on **File → Exit**.

Note:

If you subsequently rename your ControlWave project, via a **File → Save As** command, you must then re-start the Variable Extension Wizard, and save the initialization files to update the project name within the initialization files; otherwise the initialization files will still have the old project name.

Format of Initialization Files

We recommend you generate these initialization files only using the Variable Extension Wizard.

Advanced users may want to edit the initialization files manually, with an ASCII text editor. Exercise extreme care when doing this, because syntactical errors could result in problems in your ControlWave project.

Do NOT leave spaces between lines of these files.

__LISTS.INI

```
*LIST listnumber
variable1
variable2
:
variablen
```

where *listnumber* is the number used to identify this list.

variable1-n are the variables in the list.

For example:

```
*LIST 1
@GV._AI_FOR_NON_ALARMS
@GV._ALARMS_BSAP_PORT1
@GV._ALARMS_BSAP_PORT1
@GV._ALARMS_BSAP_PORT10
@GV._ALARMS_BSAP_PORT11
@GV._ALARMS_BSAP_PORT11
@GV._T16_AVG_DUR
```

__RBE.INI

```
variable1 [deadband]
variable2 [deadband]
:
variablen [deadband]
```

where:

*variable1 -
variablen*

are the names of variables you want marked for RBE collection.

[deadband]

is an RBE deadband applied to analog variables. This does NOT apply for BOOL variables. The deadband can be entered as a constant value, or a variable name can be entered, in which case, the value of that variable will serve as the deadband.

Example:

```
@GV.TANK_LEVEL @GV.TANK_LVL_DB
@GV.PRESSURE_NOW 10
```

__VAR_DEFS.INI

This file is created by the Variable Extension Wizard; however, it may be edited manually using a text editor.

```
[VERSION]
ProjName=project_name
Build=number
Date=mm/dd/yy hh:mm:ss
NumAlms=number_of_alarms

[SIG_n]
Name=variable_name
Alarm=variable_type
Units=text
Desc=description
LogPri=logical_alarm_priority
HiLimit=high_limit
HiPri=high_priority
HiHiLimit=high_high_limit
HiHiPri=high_high_priority
LoLimit=low_limit
LoPri=low_priority
LoLoLimit=low_low_limit
LoLoPri=low_low_priority
HiDB=high_deadband
LoDB=low_deadband
```

<i>project_name</i>	the name of the ControlWave Designer project.
<i>number</i>	the version number of the build for this project (incremented automatically by the Variable Extension Wizard).
<i>mm/dd/yy hh:mm:ss</i>	the date and time the project was built where mm/dd/yy refers to the 2 digit month, day, and year, respectively, and hh:mm:ss refers to the hour (0-23), minute, and seconds, respectively.
<i>number_of_alarms</i>	the number of alarms defined in the file
<i>[Sig_n]</i>	the number of the variable. A separate [Sign_n] section must be defined for each variable included in the file.
<i>variable_name</i>	the variable name, including any instance name
<i>variable_type</i>	the type of the alarm variable. This can be any one of the following:

	Analog	Analog alarm variable
	Log_True	Logical alarm variable. Alarm is generated when its value becomes TRUE.
	Log_False	Logical alarm variable. Alarm is generated when its value becomes FALSE.
	Log_State	Logical alarm variable. Alarm is generated when its value changes state from TRUE to FALSE or FALSE to TRUE.
	None	Regular non-alarm variable.
<i>text</i>		<p>For analog variables this refers to the engineering units text, e.g. MSCFH, GPM, MGD, DEGC, DEGF, etc. Units text can be up to six (6) characters. The 6 characters of units text may alternatively be held in a STRING variable, in which case the string variable name should be specified here, preceded by a tilde '~' character.</p> <p>For logical (BOOL) variables, this refers to the ON / OFF text of the variable. Up to six characters of ON text followed by a slash '/' and then up to six characters of OFF text is supported. The ON/OFF text may alternatively be held in a STRING variable, in which case the string variable name should be specified here, preceded by a tilde '~' character.</p>
<i>description</i>		Descriptive text for the variable, up to 64 characters. Alternatively, the descriptive text may be stored in a separate STRING variable, in which case the string variable name should be specified here, preceded by a tilde '~' character.
<i>high_limit, high_high_limit, low_limit, low_low_limit</i>		For Analog alarm variables ONLY. These limits specify alarm limits, for this analog alarm variable. When the variable's value exceeds the high_limit or high_high_limit, or falls below the low_limit, or low_low_limit, alarm messages are generated. When the variable's value comes back within limits, a return-to-normal message is generated. These limits may be entered as constant values, or they may be stored in variables, in which case the variable name would be entered here, preceded by a

low_deadband,
high_deadband

tilde '~'. NOTE: Deadbands may be set up to reduce fluctuations into and out-of the alarm state.

In the case of a high, or high-high alarm, the alarm condition does not clear (i.e. generate a 'return to normal' alarm message) until the value of the variable goes below the alarm limit, minus the value of the high_deadband. In the case of a low, or low-low alarm, the alarm condition does not clear until the value of the variable rises above the alarm limit, plus the value of the low_deadband. These deadbands may be entered as constant values, or they may be stored in variables, in which case the variable name would be entered here, preceded by a tilde '~'.

logical_alarm_priority
high_priority,
high_high_priority,
low_priority,
low_low_priority

Alarm priorities designate the importance of the alarm. Logical (BOOL) variables have a single alarm priority; analog alarm variables can have up to four alarm priorities – one for each alarm limit. Alarm priorities from least important to most important are:

Event (Evt)
Operator Guide (OpG)
Non Critical (NCrit)
Critical (Crit)

Example:

```
[Version]
ProjName=bktest1
Build=7
Date=11:25:16 01-21-08
NumAlms=3
[SIG_5]
Name=@GV.PRESSUR3_CURRENT
AI=1
[SIG_1]
Name=@GV.OUTPUT_1
Alarm=Log_True
LogPri=0
Units=START/STOP
[SIG_2]
Name=@GV.PRESSUR1_CURRENT
Desc=CURRENT PRESSURE READING
Units=PSI
[SIG_3]
Name=@GV.PRESSUR3_CURRENT
Alarm=Analog
HiPri=2
HiLimit=@GV.PRESSURE_HI_LIM
```

```
HiHiPri=3
HiHiLimit=@GV.PRESSURE_HIHI_LIM
LoPri=1
LoLimit=@GV.PRESSURE_LO_LIM
LoLoPri=0
LoLoLimit=@GV.PRESSURE_LOLO_LIM
LoDB=@GV.PRESSURE_LO_DB
AI=1
MI=1
```

Troubleshooting Tips

I can't see function blocks for the alarms, audit, or lists I created when I open my project?

You won't. The Variable Extension Wizard is an alternate method for creating these data structures, so they won't appear as function blocks in your project – they are configured via the INI files only.

The wizard is hung up when I type in certain fields. I can't go to another field. What should I do?

The Wizard won't let you move to another field unless you first press the **[Enter]** key. Press **[Enter]** and you should be able to proceed. The **[Esc]** also allows you to exit the field.

I ran the Variable Extension Wizard to Create My Alarms, but they don't appear in my Project?

Did you rename the project? If so, you must re-start the Variable Extension wizard in the new project, and save the initialization files again. Otherwise, they will still hold the *old* project name, before you renamed it.

Variables and Data Types

Variables are structures that hold a single numerical, Boolean, or string value, that can typically be changed or updated either by user intervention, or by logic in your ControlWave project. Variables serve the same purpose as 'signals' used in the Network 3000 series product line.

Variables can be any of several different data types. For example, a numerical variable could be of data type REAL, or INT (integer).

Global Variables Vs. Local Variables

Variables fall into one of two categories – local and global. They are declared as either local or global when you create them in a particular program. Generally, unless you have a specific reason for defining a variable as global, for example, it is an I/O variable, or you know it has to be used in more than one of the POU's listed in the 'Logical POU's' branch of the project tree, you should define it as local.

Global variables:

- may be accessed and used in any or all of the POU's in your project
- must be declared as VAR_GLOBAL in the global variables declaration worksheet
- must be declared as VAR_EXTERNAL in each POU in which they are used

Name	Type	Usage	Description	Address	Init	Retain	PDD	OPC
[-] Default								
[-] IO_GLOBAL_VARIABLES								
F101_INPUT	REAL	VAR_GLOBAL		%ID8		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
F101_INPUT_ZERO	REAL	VAR_GLOBAL		%QD0		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
F101_INPUT_SPAN	REAL	VAR_GLOBAL	Flow input	%QD4	500.000000	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
F101_OUTPUT_ZERO	REAL	VAR_GLOBAL		%QD130		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
F101_OUTPUT_SPAN	REAL	VAR_GLOBAL		%QD134	100.000000	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
F101_OUTPUT	REAL	VAR_GLOBAL		%QD138		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Data Type points to the Type column.

Variable name points to the Name column.

Usage shows whether this variable is used locally or globally points to the Usage column.

Optional description (used as a comment in the program) points to the Description column.

Internal address used to store and reference the variable points to the Address column.

Initial value of the variable points to the Init column.

Check RETAIN if you want to keep the variable in retain memory. points to the Retain column.

Check PDD if you want to collect this variable via OpenBSI utilities such as DataView. points to the PDD column.

Check OPC if you want to export this variable for use in an HMI database such as in an OpenEnterprise Server. points to the OPC column.

The Global Variables worksheet is shown, above. All variables are defined as 'VAR_GLOBAL' because they can be used in any logical POU of the project.

Local variables:

- are only used in a single POU of your project
- they are unknown to all other POUs in your project
- are declared in the worksheet of the POU in which they are used

Usage shows whether this variable is used locally or globally

Check RETAIN if you want to keep the variable in retain memory.

Check PDD if you want to collect this variable via OpenBSI utilities such as DataView.

Check OPC if you want to export this variable for use in an HMI database such as in an OpenEnterprise Server.

Variable name	Name	Data Type	Usage	Optional description (used as a comment in the program)	Address	Init	Retain	PDD	OPC
[-] Default									
	LEAD_LAG_1	LEAD_LAG	VAR	Lead Lag smooths...			<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	F101_INPUT	REAL	VAR_EXTERNAL				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	V002	REAL	VAR			0.0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	V003	REAL	VAR			0.7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	V004	BOOL	VAR			FALSE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

The Local Variables Worksheet for a particular POU in the project tree is shown, above. Notice that the F101_INPUT variable is defined as 'VAR_EXTERNAL' and several fields are NOT available. This is because F101_INPUT is a global variable which happens to be used in this POU, but is defined in the Global Variables Worksheet (see previous page).

Variable Addressing

Variables **addresses** identify where a variable is located within the ControlWave project. In general, users do NOT need to be concerned with variable addresses, as they are assigned automatically by the I/O Configurator, or when the variable is created.

Variable addresses follow the format:

AT %locationsize. Address

Where: *location* is one of the following:

- I indicates that this is a physical input (input I/O variable)
- Q indicates that this is a physical output (output I/O variable)

M indicates that this is a variable in memory (non-I/O variable)

size is one of the following

X single bit size (only used with data type of BOOL)

nothing if left blank, indicates single bit size

B byte size (8 bits)

W Word size (16 bits)

D Double word size (32 bits)

Address is the actual address in memory

An example variable address is shown below:

AT %MW 1.1018

System Variables

One special **category** of variables is **system variables**. System variables are created automatically by the system and are used for special 'housekeeping' purposes within the ControlWave system. See the *System Variables* section in this manual for more information.

Data Types

IEC 61131-3 supports three different categories of data types.

Elementary data types are used as building blocks of more complex data types. The elementary data types are shown in the table, below:

Data Type	Description	Size (in bits)	Valid Range
BOOL	Boolean	1	1 or 0 (TRUE or FALSE)
SINT	Short Integer	8	-127 ... 127
INT	Integer	16	-32768 ... 0 ... 32767
DINT	Double integer	32	-2,147,483,648 up to 2,147,483,647 ?
USINT	Unsigned short integer	8	0 up to 255
UINT	Unsigned integer	16	0 up to 65535
UDINT	Unsigned double integer	32	0 up to 4,294,967,295
REAL	Real numbers	32	$\pm 1.18 \times 10^{38}$ up to $\pm 3.40 \times 10^{38}$
TIME	Time (duration)	32	+ #4,294,976,295 milliseconds up to + #4,294,976,295 seconds
BYTE	Bit string of length 8	8	0x00...0xFF
STRING	Sequence of characters	80	
WORD	Bit string of length 16	16	0x0000 ... 0xFFFF
DWORD	Bit string of length 32	32	0x00000000 ... 0xFFFFFFFF

Generic Data Types are data types made up of elementary data types. For example, ANY_BIT, or ANY_INT.

User Defined Data Types are data types created by the user, such as ARRAYS.

Notes about STRING variables

The standard IEC62591 STRING data type allows up to 80 characters. You can also create string variables using user-defined STRING data types of varying lengths. Be aware that in either case, there are restrictions on displaying strings in programs outside of ControlWave Designer.

- ControlWave RTUs do not report strings that exceed 127 characters and behave as if the variable does not exist when data requests come in for that variable from software.
- OpenEnterprise SCADA software, OpenBSI tools such as DataView, Web_BSI web pages communicating over a serial connection, and any other program using the RDB interface to retrieve data can only display the first 64 characters of a ControlWave string variable.
- Web_BSI web pages communicating over IP can display up to 127 characters of a string variable's value.

Variable Naming Conventions

- Variable names consist of a combination of letters (A-Z, a-z), numbers (0-9) and the underscore character '_'.
- The first character of a variable name **cannot** be a number.
- Variables are **not** case sensitive, i.e. MY_VARIABLE, my_variable, and mY_vAriaBle are all considered to be the same variable name.
- Although you won't always see it, in addition to the variable name you enter, the system automatically precedes every variable by one or more instance names, separated by periods, depending upon where the variable was defined ('@GV.' for global variables, task and function block instance names for local variables) e.g. @GV.F101_INPUT or Flow1.V003
- If you have OpenBSI Utilities Version 4.0 or earlier, we recommend your variable names be limited to 20 characters or less (including the instance name or '@GV.' described above). This is recommended because prior to OpenBSI Version 4.1, tools such as DataView only recognized the first 20 characters; and so, that is the only portion of the variable name those tools will display. Newer versions recognize up to 64 characters.
- If you decide to use longer variable names (up to 128 characters are allowed), only the first 30 characters will be recognized within ControlWave Designer. If you have variables in your ControlWave POU worksheet with more than 30 characters, however, make sure there are no two variables in which the first 30 characters are the same, or else those two variables will be treated as the same variable.

For example, two variables named:

```
COMPRESSOR_STATION_FOUR_STATUS_ON
```

and

```
COMPRESSOR_STATION_FOUR_STATUS_OFF
```

should not be included in the same worksheet because the first 30 characters 'COMPRESSOR_STATION_FOUR_STATUS' are the same, and therefore the difference between the '_ON' and '_OFF' would not be recognized by the compiler.

Here are some legal variable names:

COMPRESSOR_4_STATUS

_PUMP_START

tank_level_hi_alarm

Here are some **illegal** variable names, and the reason they are illegal:

1_STATION4_MAINSWITCH (*illegal because it starts with a number*)

PUMP#4_START (*illegal because the '#' character is **not** allowed*)

Versions and Compatibility

To use ControlWave Designer with a particular ControlWave firmware revision, you must ensure that the compatible resource is loaded in ControlWave Designer. The following table lists each ControlWave hardware platform and the compatible resource required for each given firmware revision.

Controller Type	Compatible Resource	Firmware Required
ControlWave Process Automation Controller Firmware Prefix: CWP	IPC_30 IPC_33 IPC_40	Earlier than 03.10 03.10 or newer 04.40 or newer
ControlWave Low Power (LP) Controller Firmware Prefix: LPS	IPC_30 IPC_33 IPC_40	Earlier than 03.10 03.10 or newer 04.40 or newer
ControlWave Redundant Controller Firmware Prefix: CWP	IPC_30 IPC_33 IPC_40	Earlier than 03.10 03.10 or newer 04.40 or newer
ControlWave Micro Controller Firmware Prefix: CWM Firmware Prefix: CEW (33 MHz)	ARM_L_33	4.00 or newer
	ARM_L_40	4.40 or newer
ControlWave Electronic Flow Meter (EFM) Firmware Prefix: CWE	ARM_L_33 ARM_L_40	4.32 or newer 4.40 or newer
ControlWave Gas Flow Computer (GFC) Firmware Prefix: CWI	ARM_L_33 ARM_L_40	4.40 or newer 4.40 or newer
ControlWave Express GFC/RTU/PAC: 14 Mhz CPU Firmware Prefix: E1S	ARM_L_40	4.60 or newer
ControlWave Express GFC/RTU/PAC: 33 Mhz CPU Firmware Prefix: E3S	ARM_L_40	4.60 or newer
ControlWave Explosion Proof GFC (XFC) Firmware Prefix: CWX	ARM_L_40	4.41 or newer
ControlWave_10, _30, _35 Firmware Prefix: C_3	ARM_L_40	4.50 or newer (10/30) 4.70 or newer (35)
ControlWave_31 Firmware Prefix: C_1	ARM_L_40	4.70 or newer

Additionally, the ControlWave I/O Expansion Rack has a firmware prefix of **CWR** and the ControlWave Micro I/O Expansion Rack has a firmware prefix of **CMR** and the HART Interface Board (HIB) has a firmware prefix of **HBA**.

Note: Some Controller types listed above have been discontinued, though they may still be installed in the field.

ControlWave Firmware Release Summary

For full details, consult official release notes.

Firmware Release	Resource File needed in ControlWave Designer and Designer Version	ControlWave Designer Version	OpenBSI Version	Major Features
CWP 01.00.00 Released on April 20, 2001	IPC_30	3.0	4.0 or newer	<p>Communications: This version supported BSAP slave messages only. No BSAP Master Port capability is provided.</p> <p>This initial release incorporated the following functions and function blocks in the BBISFB Library:</p> <ul style="list-style-type: none"> • AGA3I • AGA8GROS • ALARM_ANALOG • ALARM_LOGICAL_OFF • ALARM_LOGICAL_ON • ALARM_STATE • ANOUT • ARCHIVE • AUDIT • AUTOADJUST • AVERAGER • COMMAND • COMPARATOR • CUSTOM • DACCUMULATOR • DEMUX • DIFFERENTIATOR • ENCODE • FUNCTION • HILOLIMITER • HILOSELECT • HSCOUNT • INTEGRATOR • LEAD_LAG • LIST010 • LIST020 • LIST030 • LIST050 • LIST100 • MUX • PDO • PID3TERM • REG_ARRAY • R_INT • R_RND • SEQUENCER • STEPPER • TOT_TRND • VLIMIT
CWP 02.00.00 LPS 02.00.00 Released on November 20, 2001	IPC_30	3.0	4.02 or newer	<p>New features include BSAP Master support, IP and BSAP client-server, generic serial protocol, and the following new function blocks:</p> <ul style="list-style-type: none"> • AGA3 • AGA8DETAIL • GENERIC_SERIAL • AGA3TERM • CLIENT • ISO5167 • AGA5 • CRC • AGA7 • FPV • REDUN_SWITCH (currently unused) • SERVER
CWP 02.01.00 LPS 02.01.00 Released on December 18, 2001	IPC_30	3.0	4.02 or newer	Maintenance – No new features
CWP 02.10.00 LPS 02.10.00 Released on April 1, 2002	IPC_30	3.0	4.1 or newer	Maintenance – No new features
CWP 02.11.00 LPS 02.11.00 Released on May 28, 2002	IPC_30	3.0	4.2 or newer	Maintenance – No New Features

Firmware Release	Resource File needed in ControlWave Designer and Designer Version	ControlWave Designer Version	OpenBSI Version	Major Features
CWP 02.20.00 LPS 02.20.00 Released on July 11, 2002	IPC_30	3.0	4.2 or newer	<p>New function blocks for file management, as well as the VIRT_PORT function block for serial connections to a terminal server:</p> <ul style="list-style-type: none"> • FILE_CLOSE • FILE_DELETE • FILE_DIR • FILE_OPEN • FILE_READ • FILE_READ_STR • FILE_WRITE • FILE_WRITE_STR • VIRT_PORT <p>In addition, this version supported the I/O Expansion Rack interface.</p>
CWP 03.00.00 LPS 03.00.00 CWR 03.00.00 Released on September 30, 2002	IPC_30	3.0	4.2 or newer	<p>First release which supports Redundancy. (CWP only) New function blocks added:</p> <ul style="list-style-type: none"> • LIST_ELEM_NAME • SCHEDULER • VAR_FETCH • VAR_SEARCH • VMUX, <p>I/O Expansion Rack released (CWR 03.00.00) – Host must have CWP 03.00.00 or newer. I/O Expansion Rack not supported with ControlWave LP, or other platforms. Only for ControlWave Process Automation Controller.</p>
CWP 03.10.00 LPS 03.10.00 CWR 03.10.00 Released on May 15, 2003	IPC_33	3.3	5.0 or newer	<p>New System Variables including:</p> <ul style="list-style-type: none"> • _pn_IMM_DIS. • _USE_ACCOL_NAME <p>Function Block Library now called 'ACCOL3'. Numerous enhancements and corrections.</p>
CWP 03.11.00 LPS 03.11.00 CWR 03.11.00 Released on June 5, 2003	IPC_33	3.3	5.0 or newer	<p>Only change from previous version is to support ControlWave FLASH upgrade. Previous 4MB of FLASH was available. Now 8MB is supported. (ControlWave only.) This change also required boot firmware to be upgraded CWB04.</p>
CWP 04.00.00 LPS04.00.00 CWM04.00.00 CWR04.00.00 Released on September 5, 2003	IPC_33 for CWP, LPS, and CWR ARM_L_33 for CWM	3.3	5.1 or newer	<p>First release of ControlWave Micro (CWM firmware). New function blocks added:</p> <ul style="list-style-type: none"> • DBLOAD • DIAL_CTRL <p>New system variables added:</p> <ul style="list-style-type: none"> • _ALARMS_PRESENT • _AUD_ALM_PRESENT • _AUD_EVT_PRESENT • _BSAP_FLAG_SENSE • _NHP_IGNORE_NRT • _NHP_IGNORE_TS • _Px_DIAL_ACTIVE • _Px_DIAL_PORT • _TOTAL_ALARMS • _TOTAL_AUD_ALARMS • _TOTAL_AUD_EVENTS • _USE_ACCOL_NAMES

Firmware Release	Resource File needed in ControlWave Designer and Designer Version	ControlWave Designer Version	OpenBSI Version	Major Features
CWP 04.01.00 LPS04.01.00 CWM04.01.00 CWR04.01.00 Released on September 10, 2003	IPC_33 for CWP, LPS, and CWR ARM_L_33 for CWM	3.3	5.1 or newer	Maintenance – No New Features
CWP 04.10.00 LPS04.10.00 CWM04.10.00 CWR04.10.00 Released on December 17, 2003	IPC_33 for CWP, LPS, and CWR ARM_L_33 for CWM	3.3	5.2 or newer	CYBOCON function block library added. AGA3DENS function block added. New system variables: <ul style="list-style-type: none"> • _AI_FOR_NON_ALMS • _ALARMS_BSAP_PORTx • _ALARMS_IBP_DESTx
CWP 04.20.00 LPS04.20.00 CWM04.20.00 CWR04.20.00 CWE04.20.00 Released on March 19, 2004	IPC_33 for CWP, LPS, and CWR ARM_L_33 for CWM and CWE	3.3	5.3 or newer	First release for ControlWave EFM (CWE). Numerous enhancements, plus: New function blocks: <ul style="list-style-type: none"> • DISPLAY • PORTCONTROL New system variables: <ul style="list-style-type: none"> • _CPU_BUSY_P1 • _EBSAP_GROUP • _INH_SYS_EVENTS • _LOCAL_ADDRESS • _NHP_ADDITIONAL_MASK • _Px_CYCLE_INT • _Px_CYCLE_TIMEO • _Px_DCD_STATE • _Px_DTR_STATE • _Px_LOCAL_PORT • _Px_AUTO_DTR
CWP 04.30.00 LPS04.30.00 CWM04.30.00 CWR04.30.00, CWE04.30.00 Released on June 10, 2004	IPC_33 for CWP, LPS, and CWR ARM_L_33 for CWM and CWE	3.3	5.3 or newer	Larger historical archive files supported.
CWP 04.31.00 LPS04.31.00 CWM04.31.00 CWR04.31.00 CWE 04.31.00 Released on August 19, 2004	IPC_33 for CWP, LPS, and CWR ARM_L_33 for CWM and CWE	3.3	5.3 or newer	Maintenance release
CWP 04.32.00 LPS04.32.00 CWM04.32.00 CWR04.32.00 CWE04.32.00 Released on October 8, 2004	IPC_33 for CWP, LPS, and CWR ARM_L_33 for CWM and CWE	3.3	5.3 or newer	Maintenance – No new features

Firmware Release	Resource File needed in ControlWave Designer and Designer Version	ControlWave Designer Version	OpenBSI Version	Major Features
CWP 04.40.00 LPS04.40.00 CWM04.40.00 CWR04.40.00 CWE 04.40.00 CWI 04.40.00 Released on February 7, 2005	IPC_40 for CWP, LPS, and CWR ARM_L_40 for CWM, CWE, and CWI	4.0	5.4 or newer	<p>First release for the ControlWave Gas Flow Computer (CWI).</p> <p>First release to support Report By Exception (RBE). Also, VSAT Slave support added, and NIST23 calculations are supported via a new library.</p> <p>Alarms can now be reported via BTCP to OpenBSI.</p> <p>EN/ENO IEC 61131 language support for CW and CW LP.</p> <p>Other protocols added include: CIP, and DNP.</p> <p>New function/function blocks:</p> <ul style="list-style-type: none"> • RBE • PORT_ATTRIB • STORAGE <p>• Trigonometric functions (ARCSIN, ARCCOS, ARCTAN) added for CW and LP platforms.</p> <p>Modifications:</p> <ul style="list-style-type: none"> • ARCHIVE function block: Mode 8 added. • Custom - Modbus Enron mode for date/time format added. <p>New system variables:</p> <ul style="list-style-type: none"> • _CW_LOAD_STR • _HEAP_BLK_FREE • _HEAP_CUR_FREE • _HEAP_RBLK_FREE • _HEAP_START_FREE • _Px_RBE_ACK_LIMIT • _Px_RBE_CLIENT_ID • _Px_RBE_ERE_COUNT • _Px_RBE_ERE_FORMAT • _Px_RBE_GO_ACT_ON_START • _S11_IO_BOARD_ID_STRING • _Px_RBE_PENDING • _Px_RBE_POOL_OVRFLW • _Px_RBE_POOL_SIZE • _Px_RBE_REPEAT_TIMEOUT • _Px_RBE_REXMIT_COUNT • _Px_RBE_RM_COUNT • _Px_RBE_RM_SINCE_ACK • _Px_RBE_STOP_RPT_MSG • _Px_RBE_USE_ACCOL_NAME • _Px_VSAT_MAX_RESP • _Px_VSAT_MIN_RESP • _Px_VSAT_UP_ACK_WAIT • _S12_IO_BOARD_ID_STRING • _S13_IO_BOARD_ID_STRING • _S14_IO_BOARD_ID_STRING
CWP 04.41.00 LPS04.41.00 CWM04.41.00 CWR04.41.00 CWE 04.41.00 CWI 04.41.00 CWX 04.41.00 Released on May 12, 2005	IPC_40 for CWP, LPS, and CWR ARM_L_40 for CWM, CWE, CWI, and CWX	4.0	5.4 Service Pack 1 and Newer	<p>Release to support XFC (ControlWave Explosion-Proof Gas Flow Computer) CWX04.41.</p> <p>Firmware for other platforms released for consistency purposes.</p> <p>Enhancements include:</p> <ul style="list-style-type: none"> • AIInet Slave Protocol • Archive Access as Data Arrays • New system variable: _ARCH_ACCESS_TYPE

ControlWave Designer Programmer's Handbook

D301426X012

February 2023

Firmware Release	Resource File needed in ControlWave Designer and Designer Version	ControlWave Designer Version	OpenBSI Version	Major Features
CWP 04.50.00 LPS04.50.00 CWM04.50.00 CWR04.50.00 CWE 04.50.00 CWI 04.50.00 CWX 04.50.00, C_304.50.00 Released on November 15, 2005	IPC_40 for CWP, LPS, and CWR ARM_L_40 for CWM, C_3, CWE, CWI, CWX	4.0	5.4 Service Pack 2 and Newer	New functions: <ul style="list-style-type: none"> • ARRAY_ANA_GET • ARRAY_ANA_SET • ARRAY_LOG_GET • ARRAY_LOG_SET • V_ATTRIB_GET • V_ATTRIB_SET Support for CW_10/CW_30 platform New system variables: <ul style="list-style-type: none"> • _OCTIME_ERROR • _APPLICATION_LOCKED • _TS_DELTA_ACCURACY • _Pn_MAX_SLAVES • _Pn_TOP_LEVEL_NODES • _Pn_DEAD_ARRAY • _Pn_DISABLE_ARRAY! • _Pn_TOTAL_NODES
CWP 04.60.00 LPS04.60.00 CWM04.60.00 CWR04.60.00 CWE 04.60.00 CWI 04.60.00 CWX 04.60.00 C_304.60.00 E1S04.60.00 E3S04.60.00 Released on May 25, 2006	IPC_40 for CWP, LPS, and CWR ARM_L_40 for CWM, CWE, CWI, CWX, C_3, E1S, and E3S.	4.5	5.5 Service Pack 1	Online editing of archive configuration and lists via TechView. First release for ControlWave Express product line. CWM and ARM-based support POU size greater than 64K. New system variables: <ul style="list-style-type: none"> • _ALARM_FORMAT • _Pn_RBE_STATE • _iPn_RBE_STATE • _Pn_PAD_FRONT • _Pn_PAD_BACK
CWP 04.62.00 LPS04.62.00 CWM04.62.00 CWR04.62.00 CWE 04.62.00 CWI 04.62.00 CWX 04.62.00 C_304.62.00 E1S04.62.00 E3S04.62.00 Released on October 12, 2006	IPC_40 for CWP, LPS, and CWR ARM_L_40 for CWM, C_3, CWE, CWI, CWX, E1S, and E3S.	4.5	5.5 Service Pack 2	Audit Trail record count added to Modbus Slave Enron Mode
CWP 04.63.00 LPS04.63.00 CWM04.63.00 CWR04.63.00 CWE 04.63.00 CWI 04.63.00 CWX 04.63.00 C_304.63.00 E1S04.63.00 E3S04.63.00 Released on December 22, 2006	IPC_40 for CWP, LPS, and CWR ARM_L_40 for CWM, C_3, CWE, CWI, CWX, E1S, and E3S.	4.5	5.6	Maintenance release

Firmware Release	Resource File needed in ControlWave Designer and Designer Version	ControlWave Designer Version	OpenBSI Version	Major Features
CWP 04.70.00 LPS04.70.00 CWM04.70.00 CWR04.70.00 CWE 04.70.00 CWI 04.70.00 CWX04.70.00 CMR04.70.00 C_104.70.00 C_304.70.00 E1S04.70.00 E3S04.70.00 Released January 17, 2007	IPC_40 for CWP, LPS, and CWR ARM_L_40 for all others	4.5	5.6	<p>New platforms: ControlWave_35 (C_3) ControlWave_31 (C_1) ControlWave Micro I/O Expansion Rack (CMR)</p> <p>New function blocks:</p> <ul style="list-style-type: none"> • BTI • TCHECK • XMTR <p>New I/O boards for expansion racks.</p> <p>Alarm System: When upgrading the firmware from 04.32 or earlier versions, collect all alarms from the running units prior to upgrading to this release as the new firmware performs a complete re-initialization of the alarm system.</p>
CWP 04.71.00 LPS04.71.00 CWM04.71.00 CWR04.71.00 CWE 04.71.00 CWI 04.71.00 CWX 04.71.00 C_304.71.00 C_104.71.00 CMR04.71.00 E1S04.71.00 E3S04.71.00 Released March 14, 2007	IPC_40 for CWP, LPS, and CWR ARM_L_40 for all others	4.5	5.6	Maintenance Release – No new features
CWP 04.72.00 LPS04.72.00 CWM04.72.00 CWR04.72.00 CWE 04.72.00 CWI 04.72.00 CWX04.72.00 C_304.72.00 C_104.72.00 CMR04.72.00 E1S04.72.00 E3S04.72.00 Released June 28, 2007	IPC_40 for CWP, LPS, and CWR ARM_L_40 for all others	4.5	5.6 Service Pack 1	Primarily a maintenance release – No new features, however, DNP3 performance was increased.

Firmware Release	Resource File needed in ControlWave Designer and Designer Version	ControlWave Designer Version	OpenBSI Version	Major Features
CWP 04.80.00 LPS04.80.00 CWM04.80.00 CWR04.80.00 CWE04.80.00 CWI04.80.00 CWX04.80.00 C_304.80.00 C_104.80.00 CMR04.80.00 E1S04.80.00 E3S04.80.00 Released September 27, 2007	IPC_40 for CWP, LPS, and CWR ARM_L_40 for all others	4.7	5.7	<p>New function blocks:</p> <ul style="list-style-type: none"> • HWSTI • TP-27 liquid FB library (LIQTAB23E, LIQTAB24E, LIQTAB53E, LIQTAB54E, LIQTAB59E, LIQTAB60E) <p>Support for HEX Repeater protocol Support for ON/OFF and units text in DISPLAY function block Support for Audit / Archive clear function in RTU (Clear History function in OpenBSI) Support for configurable timeout in Immediate Response Mode. Support for variable buffer length in Generic Serial Port Support for collecting Archive Files as if they were Data Arrays New system variables for load validation: _LOAD_MEM_PRESENT _LOAD_SRC_PRESENT _LOAD_BOOT_PRESENT _LOAD_MEM_CRC _LOAD_BOOT_CRC _LOAD_SRC_CRC</p>
CWP 04.90.00 LPS04.90.00 CWM04.90.00 CWR04.90.00 CWE 04.90.00 CWI 04.90.00 CWX04.90.00 C_304.90.00 C_104.90.00 CMR04.90.00 E1S04.90.00 E3S04.90.00 C5R04.90.00 Released May 8, 2008	IPC_40 for CWP, LPS, and CWR ARM_L_40 for all others	4.7	5.7 Service Pack 1	<p>Support for new platform: ELAN processor for ControlWave I/O Expansion Rack (C5R). Support for simplified alarm configuration using ALARM FB with the Variable Extension Wizard in ControlWave Designer. Standard application licensing. Support in RDB for program instance names in ACCOL format variables other than @GV. Support for RTU to RTU transfer of archive files. Support for Enron Modbus reading of archive files in wraparound mode to simulate the data arrays used as archive data storage by some vintage systems. Support for new I/O board in ControlWave XFC.</p> <p>New system variable: _JULIAN_TIME New Function Blocks: • ALARM</p>
CWP 04.91.00 LPS04.91.00 CWM04.91.00 CWR04.91.00 CWE 04.91.00 CWI 04.91.00 CWX04.91.00 C_304.91.00 C_104.91.00 CMR04.91.00 E1S04.91.00 E3S04.91.00 C5R04.91.00 Released June 13, 2008	IPC_40 for CWP, LPS, and CWR ARM_L_40 for all others	4.7	5.7 Service Pack 1	Maintenance Release – No new features

Firmware Release	Resource File needed in ControlWave Designer and Designer Version	ControlWave Designer Version	OpenBSI Version	Major Features
CWP 04.92.00 LPS04.92.00 CWM04.92.00 CWR04.92.00 CWE 04.92.00 CWI 04.92.00 CWX04.92.00 C_304.92.00 C_104.92.00 CMR04.92.00 E1S04.92.00 E3S04.92.00 C5R04.92.00 Released June 27, 2008	IPC_40 for CWP, LPS, and CWR ARM_L_40 for all others	4.7	5.7 Service Pack 1	Maintenance Release – No new features
CWP 05.00.00 LPS05.00.00 CWM05.00.00 CWR05.00.00 CWE 05.00.00 CWI 05.00.00 CWX05.00.00 C_305.00.00 C_105.00.00 CMR05.00.00 E1S05.00.00 E3S05.00.00 C5R05.00.00 Released April 17, 2009	IPC_40 for CWP, LPS, and CWR ARM_L_40 for all others	4.7	5.7 Service Pack 2	Support for ControlWave WXFC Support for Hart Interface Board (HIB) Support for redundant AO/DO with readback New function block: HART
CWP 05.10.00 LPS05.10.00 CWM05.10.00 CWR05.10.00 CWE 05.10.00 CWI 05.10.00 CWX05.10.00 C_305.10.00 C_105.10.00 CMR05.10.00 E1S05.10.00 E3S05.10.00 C5R05.10.00 Released September 4, 2009	IPC_40 for CWP, LPS, and CWR ARM_L_40 for all others	4.7	5.7 Service Pack 2	New function blocks: <ul style="list-style-type: none"> ● FIELDBUS ● LIQTAB59D (In liquids library) ● LIQTAB60D (In liquids library) New library: FB_RETAIN Upgrades to new standards for AGA8Gros and ISO5167 Archive files now support ASCII data. Various minor enhancements.

Firmware Release	Resource File needed in ControlWave Designer and Designer Version	ControlWave Designer Version	OpenBSI Version	Major Features
CWP 05.11.00 LPS05.11.00 CWM05.11.00 CWR05.11.00 CWE 05.11.00 CWI 05.11.00 CWX05.11.00 C_305.11.00 C_105.11.00 CMR05.11.00 E1S05.11.00 E3S05.11.00 C5R05.11.00 Released December 3, 2009	IPC_40 for CWP, LPS, and CWR ARM_L_40 for all others	4.7	5.7 Service Pack 2	New function block: BASE_DENSITY (in Liquids library) New system variable: _INH_EXTERNAL_EVENTS
CWP 05.12.00 LPS05.12.00 CWM05.12.00 CWR05.12.00 CWE 05.12.00 CWI 05.12.00 CWX05.12.00 C_305.12.00 C_105.12.00 CMR05.12.00 E1S05.12.00 E3S05.12.00 C5R05.12.00 Released December 18, 2009	IPC_40 for CWP, LPS, and CWR ARM_L_40 for all others	4.7	5.7 Service Pack 2	Maintenance Release – No new features
CWP 05.20.00 LPS05.20.00 CWM05.20.00 CWR05.20.00 CWE 05.20.00 CWI 05.20.00 CWX05.20.00 C_305.20.00 C_105.20.00 CMR05.20.00 E1S05.20.00 E3S05.20.00 C5R05.20.00 Released April 9, 2010	IPC_40 for CWP, LPS, and CWR ARM_L_40 for all others	5.0	OpenBSI 5.8	AUDIT FB enhanced to log user sign-on and sign-offs, and log out due to sign-off. ControlWave Micro now support full duplex comm. at 100MB FLASH storage area for configuration parameters increased from 64K to 128K. Enhancement to allow detection of I/O board failures. Number of ControlWave usernames increased from 32 to 240. New function blocks: <ul style="list-style-type: none">● USER_ACTIVE● USER_DEFINED New system variables: _SEC_SIGNIIN_AUD_ENA

Firmware Release	Resource File needed in ControlWave Designer and Designer Version	ControlWave Designer Version	OpenBSI Version	Major Features
				<p>_SEC_SIGIN_AUD_FTP_ENA _SEC_SIGIN_FAILURES _SEC_SIGNOFF_TMO</p> <p>Enhancements to DNP protocol to support secure authentication. Misc. enhancements to Enron Modbus protocol.</p>
CWP 05.21.00 CWM05.21.00 CWE 05.21.00 CEW05.21.00 CWX05.21.00 C_305.21.00 C_105.21.00 CMR05.21.00 E1S05.21.00 E3S05.21.00 C5R05.21.00 Released July 22, 2010	IPC_40 for CWP, ARM_L_40 for all others	5.0	OpenBSI 5.8	<p>New boot firmware to support additional memory:</p> <p>CWB08 for ControlWave PAC (CWP) and ControlWave I/O Expansion Rack (CWR) to support 32MB FLASH, and 64MB SDRAM.</p> <p>CAB0521 for ControlWave Micro (CWM) and ControlWave Micro I/O Expansion Rack (CMR) to support 16MB FLASH and 64MB SDRAM.</p> <p>CBE0521 for ControlWave EFM (CWE) to support 16MB FLASH and 64MB SDRAM. NOTE: This is for the new EFM modules; earlier versions had no SDRAM.</p> <p>Support added for read-only Modbus slave.</p>
CWP 05.30.00 CWM05.30.00 CWE 05.30.00 CEW05.30.00 CWX05.30.00 C_305.30.00 C_105.30.00 CMR05.30.00 E1S05.30.00 E3S05.30.00 C5R05.30.00 Released September 9, 2010	IPC_40 for CWP ARM_L_40 for all others	5.0	OpenBSI 5.8	<p>Support added for RS-485 serial communication to CW and CW Micro I/O expansion racks. Including system variables:</p> <p>_Pn_RESET_STATISTICS _Pn_STATISTICS_ARRAY</p> <p>Open Modbus Master / Slave custom modes now support an alternate TCP port.</p> <p>DNP protocol updated to level 3 compliance.</p>
CWP 05.40.00 CWM05.40.00 CWE 05.40.00 CEW05.40.00 CWX05.40.00 C_305.40.00 C_105.40.00 CMR05.40.00 E1S05.40.00 E3S05.40.00 C5R05.40.00 Released February 4, 2011	IPC_40 for CWP ARM_L_40 for all others	5.0	OpenBSI 5.8 Service Pack 2	<p>Note: New 33MHz CPU board for CW Express with new boot flash.</p> <p>New function block:</p> <ul style="list-style-type: none"> • AUDIT_SELECTED <p>Enhancements to XMTR function block to allow calibration of 3508 and 3808 transmitters via the HART interface board and the BTI interface board.</p>

Firmware Release	Resource File needed in ControlWave Designer and Designer Version	ControlWave Designer Version	OpenBSI Version	Major Features
CWP 05.43.00 CWM05.43.00 CWE 05.43.00 CEW05.43.00 CWX05.43.00 C_305.43.00 C_105.43.00 CMR05.43.00 E1S05.43.00 E3S05.43.00 C5R05.43.00 Released March 22, 2012	IPC_40 for CWP ARM_L_40 for all others	5.0	OpenBSI 5.8 Service Pack 2	<p>Updates to KW ProConOS 1131 operating system.</p> <p>Enhancements:</p> <p>COMMAND function block enhanced to allow double-precision floating point value for runtime accumulations.</p> <p>AUTOADJ function block now supports a new on-demand check of the sensor rotor frequency.</p> <p>New system variable for BSAP slave port and port sharing: - Pn_INH_BSAP_SLAVE</p> <p>An easier method is now supported for changing Modbus slave from RTU transmission mode to ASCII transmission mode.</p> <p>XMTR function block - enhancements for transmitter calibration.</p> <p>AGA7 function block - enhanced error reporting.</p> <p>DNP3 – enhancements for specifying the host IP address.</p>
CWP 05.50.00 CWM05.50.00 CWE 05.50.00 CEW05.50.00 CWX05.50.00 C_305.50.00 C_105.50.00 CMR05.50.00 E1S05.50.00 E3S05.50.00 C5R05.50.00 Released July 31, 2012	IPC_40 for CWP ARM_L_40 for all others	5.0	OpenBSI 5.8 Service Pack 2	<p>New function blocks:</p> <ul style="list-style-type: none"> ● IEC62591 ● AGA3SELECT
CWP 05.60.00 CWM05.60.00 CWE 05.60.00 CEW05.60.00 CWX05.60.00 C_305.60.00 C_105.60.00 CMR05.60.00 E1S05.60.00 E3S05.60.00 C5R05.60.00 Released March 6, 2014	IPC_40 for CWP ARM_L_40 for all others	5.35	5.9	<p>New function blocks:</p> <p>RBE_DISABLE WATCHDOG</p> <p>New version of NIST23 library DNP3 enhancements</p>

Firmware Release	Resource File needed in ControlWave Designer and Designer Version	ControlWave Designer Version	OpenBSI Version	Major Features
CWM05.70.00 Released August 13, 2014	ARM_L_40	5.35	5.9	IEC62591 FB enhanced to support discrete control.
CWP 05.71.00 CWM05.71.00 CWE 05.71.00 CEW05.71.00 CWX05.71.00 C_305.71.00 C_105.71.00 CMR05.71.00 E1S05.71.00 E3S05.71.00 C5R05.71.00 Released December 4, 2014	IPC_40 for CWP ARM_L_40 for all others	5.35	5.9 Service Pack 1	Added support to allow Field Tools to collect HART device information from wired and wireless HART networks.
CWP 05.72.00 CWM05.72.00 CWE 05.72.00 CEW05.72.00 CWX05.72.00 C_305.72.00 C_105.72.00 CMR05.72.00 E1S05.72.00 E3S05.72.00 C5R05.72.00 Released April 1, 2015	IPC_40 for CWP ARM_L_40 for all others	5.35	5.9 Service Pack 1	Maintenance Release – No new features
CWP 05.73.00 Released April 23, 2015	IPC_40	5.35	5.9 Service Pack 1	Maintenance Release – No new features
CWM05.74.00 Released July 1, 2015	ARM_L_40	5.35	5.9 Service Pack 1	Maintenance Release – No new features
CWP 05.75.00 CWM05.75.00 CWE 05.75.00 CEW05.75.00 CWX05.75.00 C_305.75.00 C_105.75.00 CMR05.75.00 E1S05.75.00 E3S05.75.00 C5R05.75.00 Released October 1, 2015	IPC_40 for CWP ARM_L_40 for all others	5.35	5.9 Service Pack 2	Maintenance Release – No new features

ControlWave Designer Programmer's Handbook

D301426X012

February 2023

Firmware Release	Resource File needed in ControlWave Designer and Designer Version	ControlWave Designer Version	OpenBSI Version	Major Features
CWP 05.76.00 CWM05.76.00 CWE 05.76.00 CEW05.76.00 CWX05.76.00 C_305.76.00 C_105.76.00 CMR05.76.00 E1S05.76.00 E3S05.76.00 C5R05.76.00 Released June 17, 2016	IPC_40 for CWP ARM_L_40 for all others	5.35	5.9 Service Pack 3	Support for array/list numbers greater than 255.
CWP 05.77.00 CWM05.77.00 CWE 05.77.00 CEW05.77.00 CWX05.77.00 C_305.77.00 C_105.77.00 CMR05.77.00 E1S05.77.00 E3S05.77.00 C5R05.77.00 Released April 28, 2017	IPC_40 for CWP ARM_L_40 for all others	5.35	5.9 Service Pack 3	Modification to ARCHIVE function block to support additional mode values for timestamp.
CWP 05.78.00 CWM05.78.00 CWE 05.78.00 CEW05.78.00 CWX05.78.00 C_305.78.00 C_105.78.00 CMR05.78.00 E1S05.78.00 E3S05.78.00 C5R05.78.00 Released September 22, 2017	IPC_40 for CWP ARM_L_40 for all others	5.35	5.9 Service Pack 3	Maintenance Release – No new features
CWP 05.79.00 CWM05.79.00 CMR05.79.00 E1S05.79.00 E3S05.79.00 C5R05.79.00 Released November 28, 2017	IPC_40 for CWP ARM_L_40 for all others	5.35	5.9 Service Pack 3	Maintenance Release – No new features

Firmware Release	Resource File needed in ControlWave Designer and Designer Version	ControlWave Designer Version	OpenBSI Version	Major Features
CWP 05.80.00 CWM05.80.00 CMR05.80.00 E1S05.80.00 E3S05.80.00 C5R05.80.00, HBA05.80.00 Released October 27, 2019	IPC_40 for CWP ARM_L_40 for all others	5.35	5.9 Service Pack 3	Updates to IEC62591 function block to support various Rosemount devices and several other enhancements. Note: IEC V1.20 is not backwards compatible with ControlWave firmware versions 5.79 or earlier.
CWP 05.91.00 CWM05.91.00 E1S05.91.00 E3S05.91.00 CMR05.91.00 C5R05.91.00, Released August 2020	IPC_40 for CWP ARM_L_40 for all others	5.35	5.9 Service Pack 3	Support added to allow binary inputs, counters, and analog inputs to be retained as events for DNP3 use. These events can be backed up redundantly over DNP3. Support added for DNP3 over UDP.
CWP 05.92.00 CWM05.92.00 E1S05.92.00 E3S05.92.00 CMR05.92.00 C5R05.92.00, Released December 2, 2020	IPC_40 for CWP ARM_L_40 for all others	5.35	5.9 Service Pack 3	Maintenance Release – No new features
CWP 06.00.00 CWM06.00.00 E1S06.00.00 E3S06.00.00 CMR06.00.00 C5R06.00.00, CEW06.00.00 Released June 7, 2021	IPC_40 for CWP ARM_L_40 for all others	5.35	5.9 Service Pack 3	New function blocks: AGA8_Part2 to support 2017 AGA8 chapter 2 calculations, and AGA8_GRS2017 and AGA8_DET2017 to support the 2017 version of AGA8 Part 1 calculations. Miscellaneous DNP3 updates including adding DNP3 strings (object 110) to class 0 to support static read, and adding event retention support for DNP3 strings (object 111). In addition, the DNP3 custom protocol was enhanced to allow slave session activation prior to host connection in order to remove dependence on initial host connection to activate DNP3 event generation.
CWP 06.01.00 CWM06.01.00 E1S06.01.00 E3S06.01.00 CMR06.01.00 C5R06.01.00, CEW06.01.00, Released September 15, 2021	IPC_40 for CWP ARM_L_40 for all others	5.35	5.9 Service Pack 3	Maintenance Release – No new features

Firmware Release	Resource File needed in ControlWave Designer and Designer Version	ControlWave Designer Version	OpenBSI Version	Major Features
CWP 06.02.00 CWM06.02.00 E1S06.02.00 E3S06.02.00 CMR06.02.00 C5R06.02.00, CEW06.02.00 Released December 22, 2021	IPC_40 for CWP ARM_L_40 for all others	5.35	5.9 Service Pack 3	Three new system variables added to report and manage a flash chip failure: _FLASH_FAILURE _FLASH_SECTOR_FAIL _DISABLE_FATAL_FLASH_FAIL
CWP 06.10.00 CWM06.10.00 E1S06.10.00 E3S06.10.00 CMR06.10.00 C5R06.10.00, CEW06.10.00 Released February 2023	IPC_40 for CWP ARM_L_40 for all others	5.35	5.9 Service Pack 3	Removed support for DNP3 SAV2 secure authentication. Added support for DNP3 SAV5 secure authentication. DNP3 events stored in archive files are now removed from the archive after collection by the host session.

Virtual Ports

The VIRT_PORT function block offers a way to get around limits on the actual number of physical communication ports. This is similar to the functionality for the OpenBSI Redirector.

Using the VIRT_PORT function block, users can now attach to a terminal server and send serial data out. (The terminal server could be one bought off the shelf or it could be any ControlWave, including the I/O Expansion Rack.)

The terminal server then sends the data out as serial data. This could be a way to add a BSAP Master port, for example.

To configure the ControlWave as a terminal server, you must set the CUSTOM user mode to 31.

Up to 126 virtual ports can be defined.

For full details on this subject, please refer to the online help for the VIRT_PORT function block.

VSAT Slave Port – Configuration

Important

This activity requires ControlWave Designer 4.0 (or newer), OpenBSI 5.4 (or newer), and 04.40 ControlWave Firmware (or newer).

Very Small Aperture Terminal (VSAT) is a protocol used with satellite communications. The VSAT slave port allows a ControlWave-series controller to be part of a VSAT system.

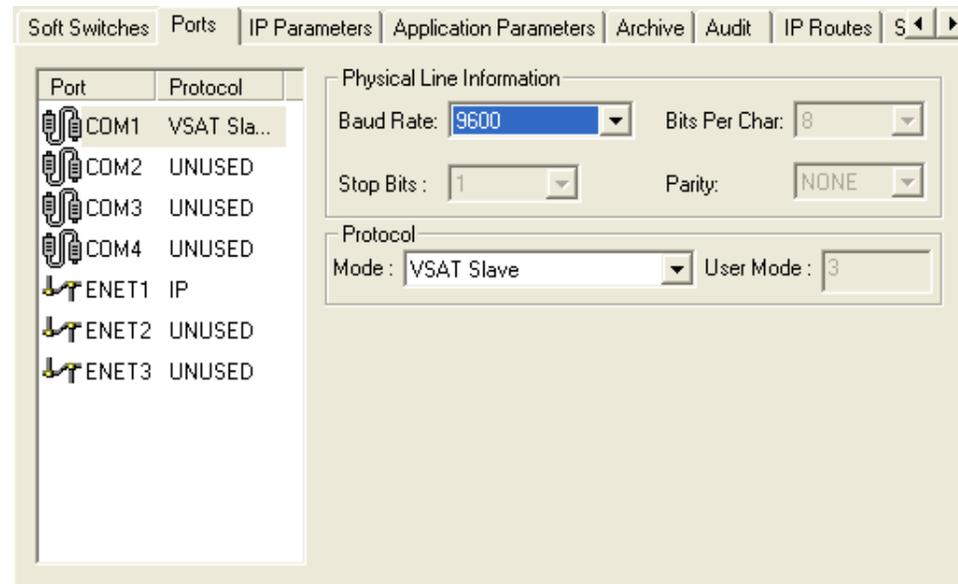
VSAT Slave Port configuration has two components:

- Configuring flash parameters
- Configuring system variables for the port

Configuring Flash Parameters

1. In the Flash Configuration Utility, click on the **Ports** tab and choose the ControlWave serial port you want to configure (COM1, COM2, etc.) Then select **VSAT Slave** as the **Mode**.

Enter the baud rate for the communication line in the "**Baud Rate**" field. The default is 9600. The maximum supported for VSAT Slave is 57600.



2. Click on the **[Save to Rtu]** button, and respond to the sign-on prompts.
3. Turn off the ControlWave, then turn it back on for the new port definition to come into effect.

Configuring System Variables

Within ControlWave Designer, start the System Variable Wizard by clicking on **View → System Variable Wizard**.

First, check the box of the port you want to configure.

Next, click on the "Configuration" button for that port.

The screenshot displays the 'Port Detail' tab of the System Variable Wizard. It features a grid of port configuration options for ports one through eleven. Each port entry includes an 'Enable' checkbox, a 'Configuration' button, an 'Information' button, and an 'RBE' button. Port One is selected, with its 'Enable' checkbox checked and its 'Configuration' button highlighted. The interface also shows a top navigation bar with tabs for IP RBE Configuration, Data Line Monitor, Redundancy, Load Validation, System Configuration, System Information, System Statistics, Alarms, System Flags, ID Strings, System Date/Time, Task Info, Port - Globals, and Port Detail.

When the wizard has successfully established communications with ControlWave Designer, and your project is open, do the following:

1. Choose the 'Port Detail' tab.
2. Select the "Enable" box for the port which will serve as the BSAP Slave.
3. Click [Configuration].
4. In the Configuration page, select only the items shown in the figure, on the next page, and enter appropriate values. A discussion of the various items appears, below:

VSAT – Minimum Response Time
(_Px_VSAT_MIN_RESP)

This defines the minimum period of time (in milliseconds) during which this VSAT Slave node will wait before responding to a message from its VSAT Master.

The purpose of this delay is to allow the VSAT Master enough time to send messages to multiple VSAT slaves before having to handle their response messages.

VSAT – Maximum Response Time (_Px_VSAT_MAX_RESP)

This defines the maximum period of time (in milliseconds) that this VSAT Slave node will wait before responding to a message from its VSAT Master. The purpose of this delay is to allow enough time for requested data to become available. If alarms become available, they will be sent immediately, regardless of this value. If no data becomes available by the conclusion of this period, an acknowledgment will be sent to the VSAT Master.

Click [OK] when finished.

The screenshot shows the 'Port 1 Configuration' dialog box with several sections. The 'Slave' section is highlighted, showing the 'VSAT - Minimum Response Time' and 'VSAT - Maximum Response Time' options checked. The 'VSAT - Maximum Response Time' field is set to 0. A callout box points to this field with the text: 'Once you select an item, you can specify the value in the corresponding field.' Another callout box points to the 'VSAT - Minimum Response Time' and 'VSAT - Maximum Response Time' fields with the text: 'These fields apply to VSAT Slave Ports.' The 'Common' section includes options like 'Poll Time' (5), 'Write Delay' (0), 'Write (CTS) Timeout' (2500), 'Ignore Echo Data' (FALSE), 'Port Supports Dial' (FALSE), 'Port performs Auto DTR shutdown', 'BSAP Pad Front', and 'BSAP Pad Back'. The 'Master' section includes 'Retries', 'Data Link Timeout', 'Idle Polling', and 'VSAT - Up Ack Wait'. The 'EBSAP Master' section includes 'Max Slaves Under Virtual Nodes', 'Top Level Nodes', 'Total Slaves On Port', 'Array Number of Dead Array', and 'Array Number of Disable Array'. The dialog box has 'OK' and 'Cancel' buttons at the bottom right.

VSAT Master Ports

OpenBSI Workstations with BSAP Master ports can serve as VSAT Masters because they understand VSAT messages. The only configuration required is to configure the UpAckDelay parameter in the BSBSAP.INI file. Similarly, ControlWave BSAP Master ports can serve as VSAT Masters. The only configuration involves setting the _VSAT_UP_ACK_WAIT system variable (shown under 'Master' in the figure above).

These parameters (UpAckDelay or _VSAT_UP_ACK_WAIT) define an interval of time (in milliseconds) during which the driver will wait after sending an ACK for a message sent by the slave. Since the VSAT slave will not be responding to the ACK, it allows the VSAT Slave time to get ready for the next request from the VSAT Master.

Web Pages – Notes About Using

The standard set of web pages, referred to as *Web_BSI*, is described in detail in the *Web_BSI Manual* (part number D301418X012). Various configuration details related to web page setup are repeated here.

Specifying the Location of Your Web Browser

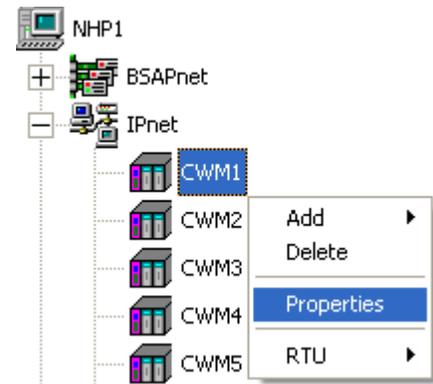
If the path of your web browser is other than the default (`\Program Files\Internet Explorer\`) you need to use a text editor to edit the `WEB_BROWSER_PATH` parameter in your NDF file to reflect the web browser's location.

```
WEB_BROWSER_PATH=C:\Program Files\Internet Explorer
```

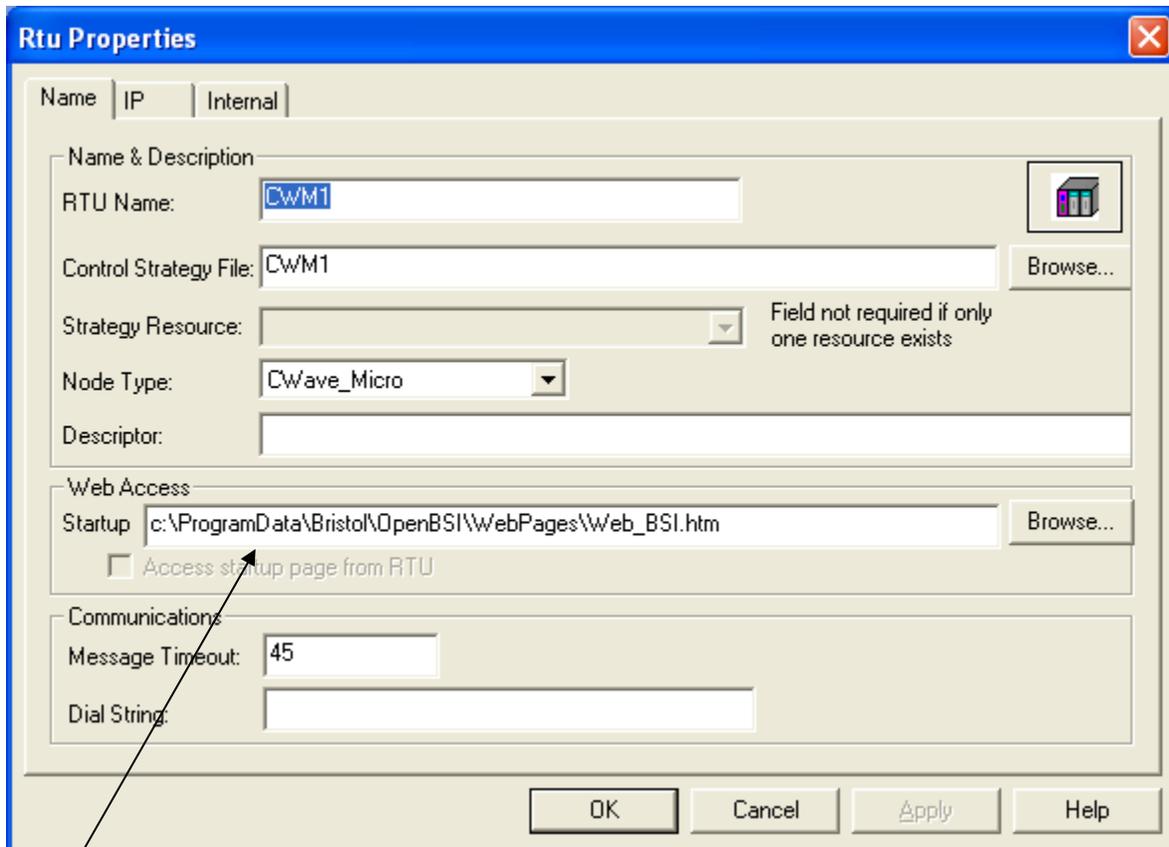
Specifying the Startup Web Page For A Controller

During OpenBSI system configuration, you must specify a startup HTML web page for each controller. This can be done in NetView's RTU Wizard when you initially add the controller or from the RTU's Properties dialog box, after the RTU is already in the network.

To access the RTU Properties dialog box, *right* click on the icon for the controller, and choose **"Properties"** from the pop-up menu.



- The startup web page resides on the PC workstation, so a full path and filename must be entered in the **"Startup"** field of the RTU Properties dialog box.
- If you would like access to the standard web page set, specify `web_bsi.htm` as the startup page. This web page is referred to as the **Main Menu**, and contains links to all of the standard web pages.



Specify the complete path and filename on the PC, of the startup web page here. (Click Browse to locate the web page, if necessary.)

Other Notes About Using Web Pages

- For optimum results, set screen resolution to 1024 x 768 when using the Web_BSI web pages.
- You can have multiple web pages open simultaneously, for example, to look at different types of data from the same RTU. To do this, just open a new instance of Internet Explorer (or open a new window in IE using the **File**→**New** command). Note, however, that if you terminate one instance (or window) communicating with a particular RTU, you will terminate *all instances or windows* communicating with that same RTU.
- If your ControlWave controller is part of a BSAP network, it will be treated as a BSAP controller; and only those configuration facilities and features available for a BSAP controller will be available.
- The standard set of web pages (Web_BSI) are stored in the directory:

`\ProgramData\Bristol\OpenBSI\WebPages`

where *openbsi_installation_path* is whatever directory you chose for the installation of OpenBSI. The default is OpenBSI.

Calling Up Web_BSI Pages

There are two different methods for calling up the Web_BSI web pages:

Important

If this is the first time you are calling up the Web_BSI web pages, you will need to use the Node Locator page, to identify the nodes with which you want to communicate. After that, you should not need to use it again, unless you are communicating with different nodes, or if your network configuration has changed.

Method 1

With NetView or LocalView running, click as follows:

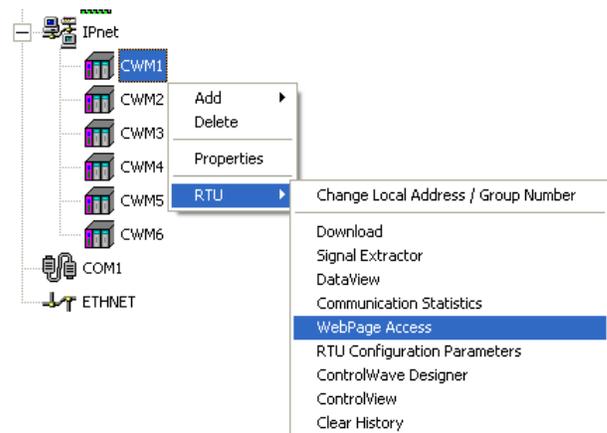
Start→Programs→OpenBSI Tools→Web Page Access→Standard Pages

Important

Depending upon what version of the Windows operating system you use, you may need to log in with Administrative privileges in order to use certain configuration web pages, in particular the Node Locator.

Method 2

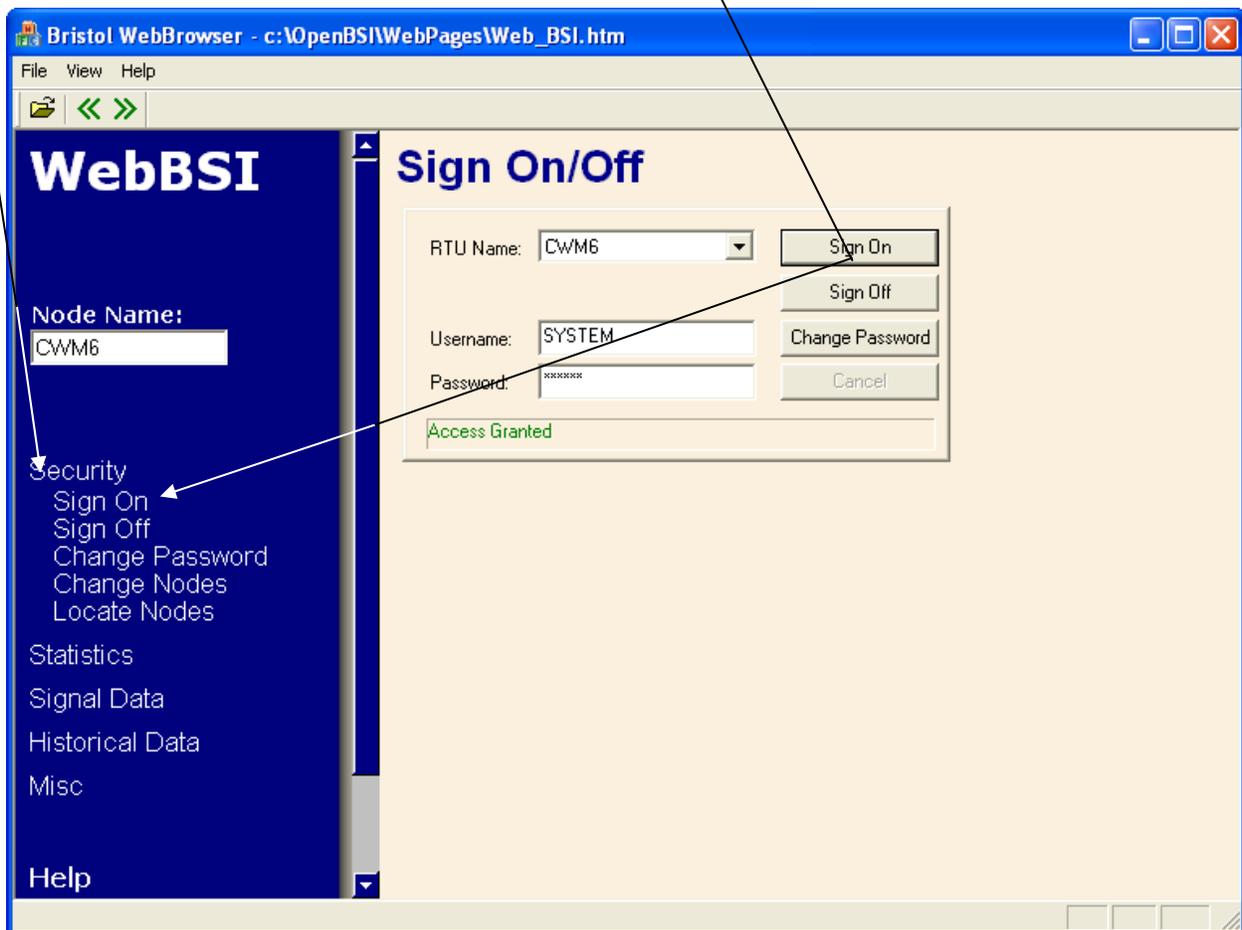
In order to call up the web page(s) associated with a particular controller, right click on the icon for the controller in the OpenBSI NetView tree, and choose **RTU→WebPage Access** from the pop-up menu. Internet Explorer will be started, and whichever startup web page associated with the controller will be displayed.



Note:

If a web page is initially being stored within the controller, you must retrieve it for viewing using the ControlView utility (RTU → **ControlView**). See the *BSI_Config User's Guide* (part number D301428X012) for information on ControlView.

To call up other web pages, click on a category, and then select from the choices that appear below it.



The Main Menu page in the standard set is shown above, although the startup page for *your* controller may be different. Typically, the Security Sign-On always appears on the Main Menu page.

The various web pages include category buttons along the left hand side, for calling up additional pages; when you click your cursor on a category button, a list of pages belonging in that category will appear below it. The category buttons are named Security, Configuration, Statistics, Signal Data, and Historical Data.

A **"Node Name"** field displays the name of the current controller from which data is being viewed on the web page.

Creating Your Own Web Pages to Use with the ControlWave

If desired, you can develop your own customized web pages to collect and display data from a ControlWave series controller. To do this, you must use ActiveX controls. See *Appendix A* of the *Web_BSI Manual* (part number D301418X012) for more information.

Appendix A: Troubleshooting Tips

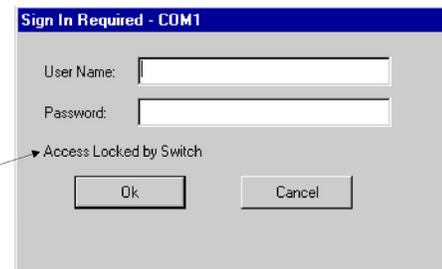
I had a previous version of ControlWave Designer installed. Now, I just installed a new copy of ControlWave Designer 3.3 (or newer), but it will only operate in demo mode. How come?

Prior to ControlWave Designer Version 3.3, a software copy protection key (dongle) had to be plugged into the parallel port of the PC. Beginning with Version 3.3, you should NOT use the copy protected key. Unplug it, and try starting ControlWave Designer again, and all functions should be available.

When I try to download a project to the ControlWave, I get an 'Access locked by Switch' message in the Sign On dialog box. What does that mean?

This typically refers to the key switch which is just above COM port 1 on the ControlWave. In order to download a project, this switch must be either in the 'REMOTE' or 'LOCAL' position, depending upon whether you are communicating serially, or using TCP/IP. (Serial communications from ControlWave Designer require the switch be in the 'LOCAL' position; TCP/IP communications support either 'REMOTE' or 'LOCAL'.)

OpenBSI downloads can occur with the switch in either the 'REMOTE' or 'LOCAL' position.



If you see this message, check to see that the key switch on the ControlWave is in the 'RUN' position. If it is, you must turn it to either 'REMOTE' or 'LOCAL'.

I am able to connect to the ControlWave, but Internet Explorer returns a '-404 File Not Found' error when I try to call up a web page.

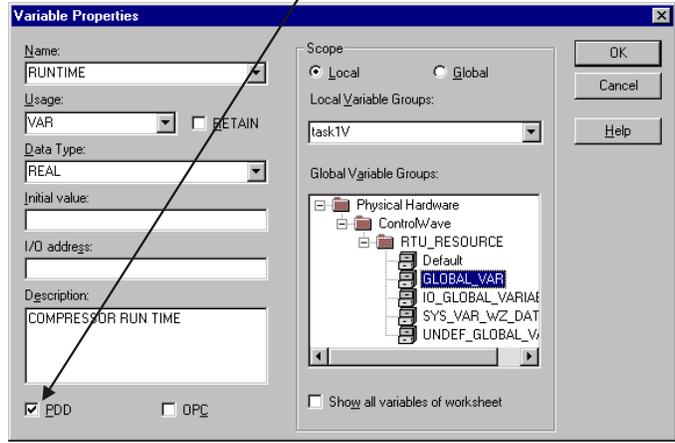
If you are using web pages, make sure you have the correct path and filename.

In DataView, when I try to do a signal search, I can't see any of the variables in the ControlWave project. Why?

DataView can only collect variables which have been defined as "PDD".

Variables which are explicitly marked as "PDD" by the user (see picture at right) can be collected by DataView if the "Marked Variables" for PDD option is selected in the Resource Settings dialog box (see below)

A variable is marked as "PDD" here.

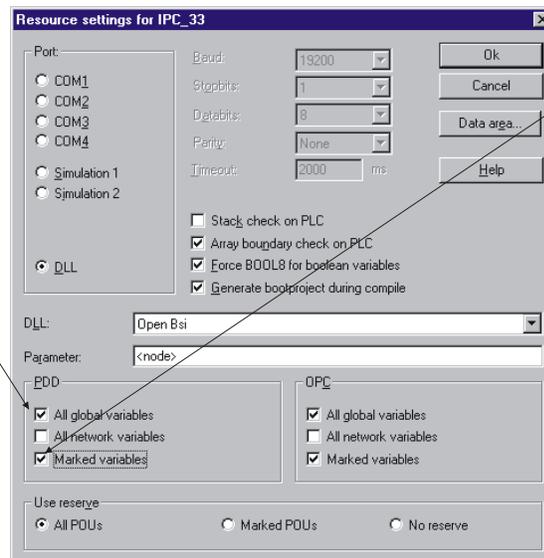


In addition, you can also choose to automatically mark all global variables as "PDD" by choosing the "All global variables" option in the Resource Settings dialog box.

Most users choose NOT to do this, since it means that all global variables will be collected, many of which are likely to not be of interest to the average user.

When this is checked, all global variables will be declared as "PDD".

When this is checked, local variables marked as "PDD" will be declared as "PDD".



In DataView, how come I can't always call up the signal lists I want to see?

In Version 5.9 Service Pack 2 and earlier, DataView could only display signal lists numbered from 1 to 255; any higher number list could not be displayed. Edit the `iListNumber` value for any LIST function blocks to be from 1 to 255.

I made changes to configuration parameters in the ControlWave (port type, user accounts, etc.) but the old settings are still in effect. How come?

This is one of the most common occurrences in ControlWave. For new settings to take effect, you must first reset the unit (turn the unit off, then turn it back on). The other reason this can occur is if you still have the default switch in the OFF position. Changes to soft switches are ignored when the default switch is OFF. On a standard ControlWave, the default switch is SW1-3, on the LP, it is SW4-3, and on the MICRO, it is SW2-3.

Communications with the ControlWave operate fine until I connect it to a Network 3000 controller, then communications are seriously degraded, or stop entirely. Why?

This has been known to happen if you are using the wrong cable type, or if the ControlWave or Network 3000 controllers are not properly grounded. See the hardware manual for details about cabling and grounding.

I tried to start ControlWave Designer to communicate with the ControlWave, but I got the message 'Could not attach to serial port'. What causes that?

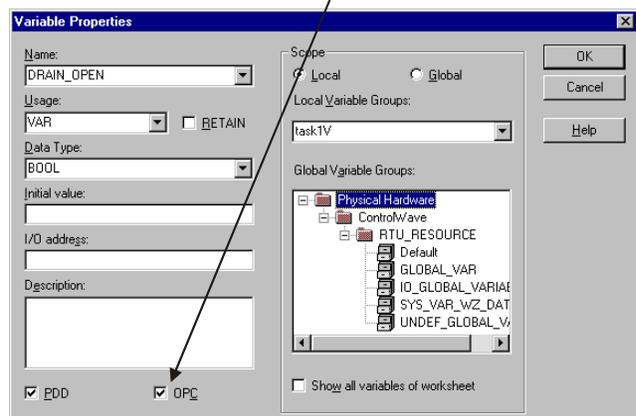
This can occur if the serial port on the PC is already being used by some other program. For example, if you are running NetView to communicate with the ControlWave, you cannot use the same PC port simultaneously to communicate directly using ControlWave Designer. You can, however, start ControlWave Designer from within NetView.

When I try to use the OPC Server to get data out of the ControlWave, I can't find anything. How come?

The OPC Server can only access variables which have been specified for OPC access.

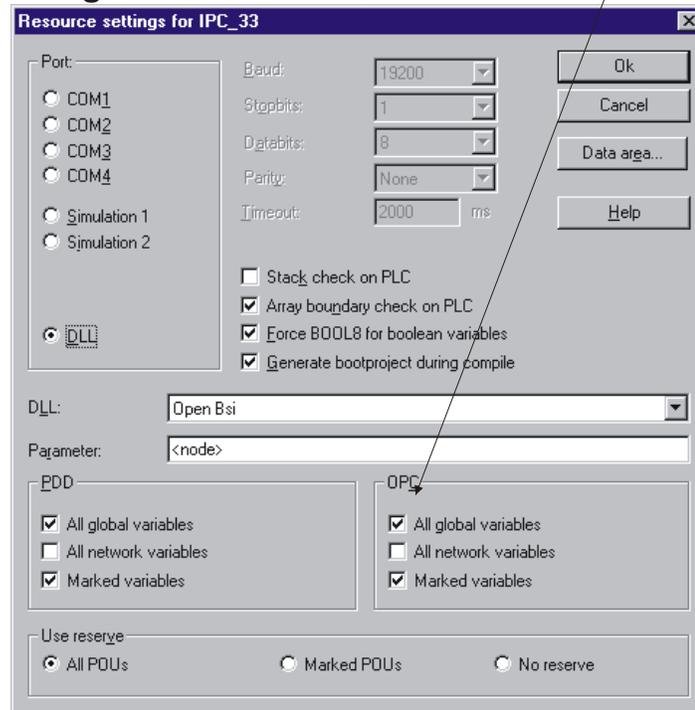
Variables which are explicitly marked as **"OPC"** by the user (see picture at right) are available to the OPC Server if the **"Marked Variables"** for OPC option is selected in the Resource Settings dialog box (see next page).

Check the **"OPC"** box



In addition, you can choose to automatically mark all global variables to be marked for **"OPC"** access by choosing the **"All global variables"** OPC option in the Resource Settings dialog box.

Make sure you have checked
"Marked variables", or
"All global variables", or both.



If you still cannot collect variables via OPC, you should check that your configuration settings for the OpenBSI Signal Extractor are correct. See Chapter 12 of the *OpenBSI Utilities Manual* for details.

In ControlWave Designer, I can't see the units text I entered for my variables. What's wrong?

If these are retain variables, you might not have enough retain memory allocated. See *Memory Usage*.

Windows isn't letting me run ControlWave Designer or the I/O Simulator. Why not?

Data Execution Settings (DEP) which exist in certain Windows versions can prevent ControlWave Designer and I/O Simulator from running. You must change the settings from Windows Control Panel. Call-up sequences vary slightly based on the operating system, but the basic sequence is:

1. Click **Start** → **Settings** → **Control Panel** → **System**
2. Click **Advanced**
3. Click **Performance** → **Settings**
4. Click **Data Execution Prevention** on **Performance Options**:
5. Do the following, based on the platform:

Windows 2003 Server or Windows 2008 Server	Windows XP or Windows 7
<i>Either:</i> Turn on DEP for essential programs and services only <i>Or</i> Turn on DEP for all programs and services except those I select: C:\windows\system32\rundll32.exe.	Turn on DEP for essential programs and services only

Using the Debug Information Tool

(ADVANCED USERS ONLY 04.40 Firmware or NEWER)

It is possible to view internal details about a ControlWave unit's operation using the ControlWave Debug Information tool. This tool is intended solely for use by Emerson Development, Engineering, and Technical/Application Support personnel, or by customers being directly assisted by these personnel.

The ControlWave Debug Information tool retrieves information such as:

- Contents of internal memory
- Contents of user stack (memory)
- Contents of messages sent/received via a particular port (similar to a Data Line Monitor)
- RBE information

Starting the Debug Information Tool:

Note:

Communications via NetView, LocalView, or TechView must be active in order to use the Debug Information Tool.

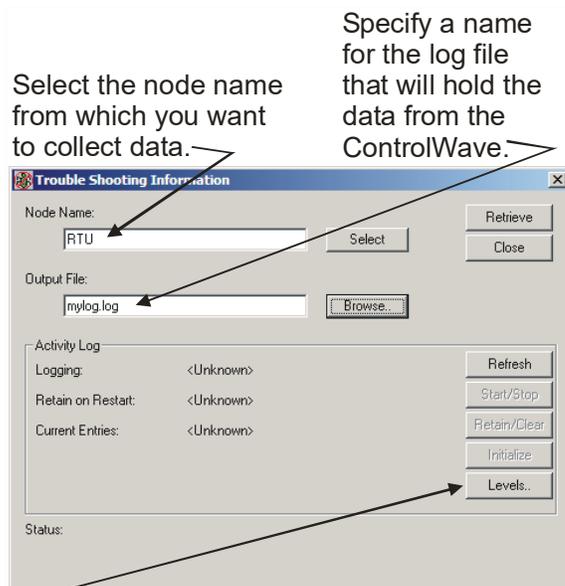
To start the Debug Information Tool, click on as follows:

Start → Programs → OpenBSI Tools → Utility Programs → ControlWave Debug Info

The Trouble Shooting Information dialog box will appear:

First, choose the ControlWave controller from which you want to collect data. Enter the name of the ControlWave in the **"Node Name"** or choose the name from the list of nodes in your NETDEF file using the **[Select]** button. (If you've used the tool before, the last node you looked at will appear in the field, by default.)

Next, provide a name for the file that will hold the collected data in the **"Output File"** field, or use the **[Browse]** file to locate one. Output files must have the extension of *.LOG. (If you've used the tool before, the last log file you



Click on **[Levels]** to specify what level of detail of data you want to collect.

used will appear in the field, by default.)

Now, you need to specify what level of data you want to collect. To do this click on the **[Levels]** button.

Specifying the Logging Levels for the Debug Information Tool

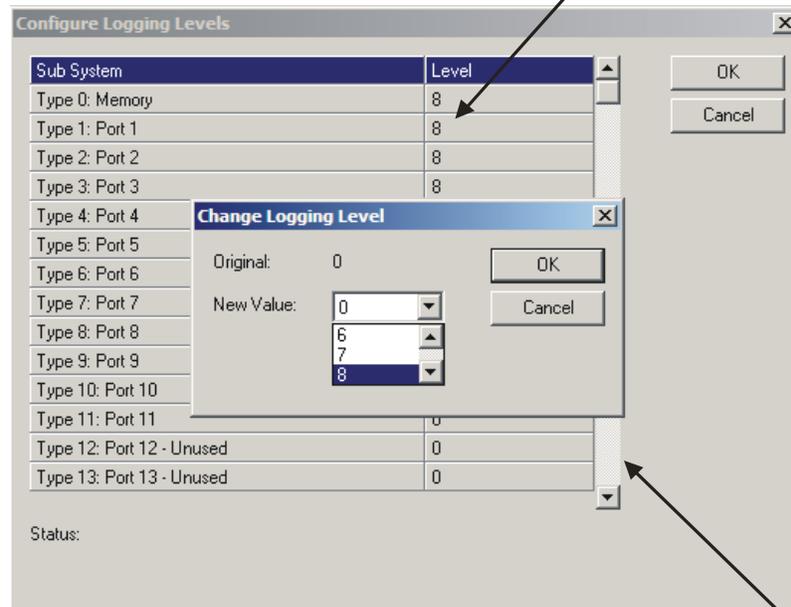
In the Configure Logging Levels dialog box, choose the sub-system (port, memory, etc.) that you're interested in, and click on the corresponding level column number.

The sub-systems are identified by a **"Type"** number. The type numbers are:

Type	Sub-System
0	Memory
1	Port 1
2	Port 2
3	Port 3
4	Port 4
5	Port 5
6	Port 6
7	Port 7
8	Port 8
9	Port 9
10	Port 10
11	Port 11
12	Port 12 – UNUSED
13	Port 13 – UNUSED
14	Port 14 – UNUSED
15	Port 15 - UNUSED
16	Virtual Ports
17	Flash Access (covers read/write of files from FLASH memory)
18	Time Synch
19	Temporary Use
20	Custom Protocol
21	CIP Protocol
22	AMOCAMS AI Net Custom Protocol
40	RBE System
41	Display System
60	Dynamic IP Routes

The Change Logging Level dialog box will appear. Logging levels range from 1 to 8. The higher the logging level, the more information will be collected, so generally, you should choose '8'. Click on **[OK]** to update the level. Repeat this process for each additional item you want to monitor. When you are finished, click on **[OK]** to exit the Configure Logging Levels dialog box.

Choose the sub-systems within the ControlWave that you want to monitor by clicking on the corresponding level.



Use the scroll bar to bring more items into view.

The level settings are saved in the ControlWave, and will be used for all subsequent logging sessions, unless changed or erased by a system cold start.

Collecting the Debug Data

Once you've finished defining the logging level, click on the **[Refresh]** button. The **[Start/Stop]** button should now be labeled **[Start]**.

Click on **[Start]** and data on the items you selected will begin to be stored in unused areas of static memory. The debug data will not affect memory needed for system operation.

If you want the data to be preserved across warm starts of the ControlWave, click on **[Retain]**.

Note:

Once the tool is started, you do not need to leave it running; since collection occurs in the background.

Viewing the Debug Data

When you decide you want to retrieve the debug data into a log file, for viewing, click on the **[Retrieve]** button in the Debug Information Tool, and the data will be stored in the log file specified.

Firmware Version: CWM 04.60.10

Link Date: 03/29

Crash Blocks: 0

**** Registers ****

*** Enhanced Crash Information. ***

Version: 0000 Complement: 0000

Mult Crash: 0

Executing:

Free Vol: 0 (0 bytes)

Free Non Vol: 0 (0 bytes)

Free Malloc: 0 bytes

*** User Stack ***

*** System Stack ***

*** Debug Activity Log ***

Retain Log: 1

Entries: 1126

Log Active: T000: 8 T001: 8 T002: 8 T003: 8 T004: 8

T005: 8

064cd064 MEM_SYS: Free 276, Address: 60006a70, Caller: 00000000

064cd158 MEM_SYS: Get 276, Address: 60006a70, Caller: 10052d04

064cd158 MEM_SYS: Free 276, Address: 60003008, Caller: 00000000

064cd3c8 P3: Lengths: Read: 11, Write: 0

064cd3c8 P3: RAW: 1002 02f58510 10100327

064cd3c8 P3: RAW: ad00 00000000 00000000

064cd408 P3: Lengths: Read: 11, Write: 0

064cd408 P3: RAW: 1002 03f68510 101003ae

064cd408 P3: RAW: 8300 00000000 00000000

064cd448 P3: Lengths: Read: 11, Write: 0

064cd448 P3: RAW: 1002 06f78510 10100341

064cd448 P3: RAW: b900 00000000 00000000

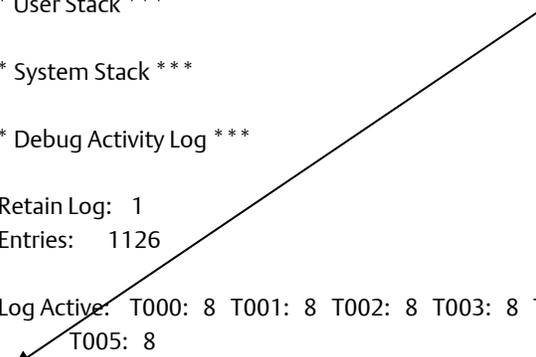
064cd448 P3: BSAP Recv, Slave/Len: 0605, S/DMex/ID: 89850310, S/D GLAD: 00000000

064cd448 P3: BSAP Send, Slave/Len: 0007, S/DMex/ID: 01870006, S/D GLAD: 00000000

064cd448 P3: Lengths: Read: 0, Write: 12

064cd448 P3: RAW: 1002 00f78706 00011003

*Timestamp in hexadecimal
format, based on system time
from day 0.*



You can view the log file in any text editor. As the information in the log file is intended primarily for Emerson Development Engineering personnel, instructions for interpreting the contents of the log file is beyond the scope of this manual.

```
DEBUG_LOG_NULL;0;0;0;NULL Entry
DEBUG_LOG_MEM_SUM;0;0;1;MEM_SYS: Sum - Ret Pages:%04x, Page Pool: %08x, Total: %08x
DEBUG_LOG_MEM_PFRET;0;1;0;MEM_SYS: Retain Page Free, %d Pages, Address: %08x
DEBUG_LOG_MEM_PARET;0;1;1;MEM_SYS: Retain Page Alloc, %d Pages, Address: %08x
DEBUG_LOG_MEM_PF;0;1;2;MEM_SYS: Page Free, %d Pages, Address: %08x
DEBUG_LOG_MEM_PA;0;1;3;MEM_SYS: Page Alloc, %d Pages, Address: %08x
DEBUG_LOG_MEM_GET;0;2;0;MEM_SYS: Get %5d, Address: %08x, Caller: %08x
DEBUG_LOG_MEM_FREE;0;2;1;MEM_SYS: Free %5d, Address: %08x, Caller: %08x
DEBUG_LOG_P1_MOD_CHANGE;1;0;0;P1: Modem Control Change: %d
DEBUG_LOG_P1_DCD_CHANGE;1;0;1;P1: Carrier Changed to: %d
```

Erasing Debug Data

If you don't want the log entries retained between warm starts of the ControlWave, click **[Clear]**, and they will be erased at the next warm start. (An exception to this is if the unit crashes and restarts; the information will not be erased.)

If you want to clear the log entries from static memory, immediately, and not wait until a warm start, click **[Initialize]**.

Other Debugging Tools (BTCP Spy, DLM Monitor)

These tools are reserved for Emerson Energy and Transportation Solutions support and development personnel and are not intended for customer use.

Appendix B: ControlWave Designer Compatibility Issues

If you install ControlWave Designer on your PC Workstation, and you already have an *earlier* version of ControlWave Designer installed, the earlier version is automatically overwritten.

Important

ControlWave Designer is backward compatible such that *older* ControlWave projects can be opened in a new version of ControlWave Designer. Any ControlWave project created or modified with the newer version of ControlWave Designer, however, cannot subsequently be brought back into the older version of ControlWave Designer.

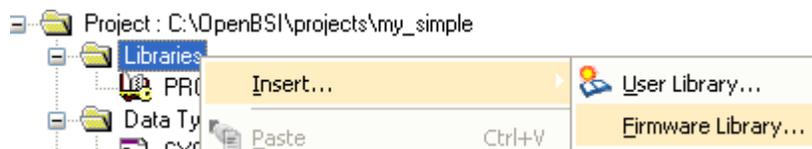
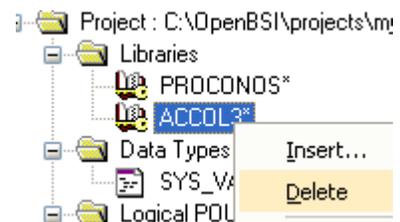
In other words, once you bring a project created with an older version of ControlWave Designer into a newer version of ControlWave Designer, you can't edit it with the older ControlWave Designer, because it would now be incompatible. Similarly, if you create an all-new project in the newer ControlWave Designer, it also cannot be used within the older ControlWave Designer.

For this reason, you should only open a project in the newer version of ControlWave Designer if you intend to edit it, from that time onwards, in the newer version.

Bringing an Older ControlWave Project into a Newer Version of ControlWave Designer

If you have an older ControlWave Designer project, that you want to open it in a newer version of ControlWave Designer, there are certain steps you must take.

1. After you have brought the project in, you must delete the ACCOL3 firmware library, and any other (*.FWL) libraries.
2. Now, insert the most recent *.FWL libraries (which come with the new version of ControlWave Designer, you are using). Firmware libraries are located in the path `\Openbsi\mwt\plc\fw_lib\`



3. Finally, re-build any user libraries (*.MWT) built based on the older libraries, by clicking on **Build** → **Make**.

Warning - I/O Configurator and Multiple Copies of ControlWave Designer

If you intend to run multiple copies of ControlWave Designer simultaneously, do NOT attempt to run multiple copies of the I/O Configurator. If you do, you risk corrupting your I/O definitions.

Index

_TIME system variables.....	375
ACCOL names	376, 377
Addressing	
in an IP network	293
in BSAP networks	52
Alarms	
_ALARM_FORMAT system variable	377
configuring	9
marking alarm variables in Variable Extension Wizard	408
priority of.....	10
Application	
licensing of standard ControlWave applications	23
parameters	27
Archive File	
configuring	29
Array	
configuring an	41
Audit	
configuration.....	43
breakpoint	
clearing a	111
setting a	110
BSAP	
local address.....	52
Master Port.....	55
Slave Port	61
what is it?	49
Challenge Handshaking Authentication Protocol (CHAP)	305
CHAP.....	355
setting the default username for	305
specifying for a PPP port	310
Cold start	
application	325
starting with a System task.....	365
system.....	324
Communication Ports	67
defaults for	81
setting up a BSAP Master Port	55
setting up a BSAP Slave Port.....	61
setting up a Modbus Port	341
setting up a PPPort	309
setting up a VSAT Slave Port	447
setting up an Ethernet Port.....	307
sharing	92
Compiling your project	95
Conditional logic	
in your ControlWave project.....	97
Configuring	
Modbus.....	335
Data Types	
IEC 61131-3.....	425
DataView	
using with ControlWave	101
Debugging your project	105
Default settings	
for communication ports.....	81
DNP3 SAV5.....	361
Downloading your ControlWave project	113
EBSAP	127
group number	52
Ethernet Port	307
Firmware libraries	315
Flash Configuration utility	
starting	149
Flash File access utility.....	155
Forcing a variable's value.....	109
Function block	
creating.....	159
executing once at startup	97
list of function blocks in ACCOLIII library	5
parameter name prefixes.....	165
Global variables.....	423
Group number	
in an EBSAP network.....	52
Historical data.....	167
I/O Configurator.....	169
I/O Mapping.....	197
I/O Simulator	283
Initialization Files	
in Variable Extension Wizard.....	416
IP addressing.....	293
IP Parameters.....	301
IP Ports	

Ethernet	307	Secure gateway.....	362
PPP	309	Security.....	345
IP Routes	311	setting the default username for CHAP protocol	
Libraries		usage.....	305
firmware and user.....	315	Security Protocols	355
Licensing		DNP3 SAV5	361
of standard ControlWave applications	23	Serial port	
Lists		sharing	92
defining in Variable Extension Wizard	410	Sharing	
Local address		a serial port	92
in a BSAP network.....	52	Slave Port.....	See BSAP Slave Port
Local variables	424	Specifying IP address for an NHP	301
Master Port	See BSAP Master Port	String Variables	426
Memory	319	System task.....	365
Flash File Access utility	155	System Variables.....	367
static memory area.....	397	User Security Management Tool.....	345
Modbus		User-created libraries	315
configuring.....	335	Variable Extension Wizard	403
exporting security data to	351	Variables	423
Modus Port.....	341	addresses of	424
Network Host PC (NHP).....	301	default variable names for I/O.....	197
Overwriting a variable's value	109	forcing or overwriting a value	109
PAP	355	naming conventions for.....	427
specifying for a PPP port	310	system	367
Passwords	345	Versions	
CHAP usage	305	compatibility of different software and firmware	
Patch POU	108	429
Point-to-Point Protocol (PPP) Port	309	Virtual Ports	445
Ports	67	VSAT Slave Port	
defaults for	81	setting up a	447
setting up a BSAP Master Port	55	Warm start	
setting up a BSAP Slave Port.....	61	application	
setting up a Modbus Port.....	341	starting with a System task	365
setting up a PPP Port.....	309	system	324
setting up a VSAT Slave Port	447	Watch window	105
setting up an Ethernet Port	307	Watchdog	
sharing	92	what happens to memory.....	323
POU size	330	Web browser	
RBE collection		specifying the location of.....	451
marking variables in Variable Extension Wizard		Web pages	
.....	407	notes about using.....	451
Resetting the ControlWave.....	343	Wizard	
Routing Internet Protocol (RIP).....	302	Variable Extension	403

ControlWave Designer Programmer's Handbook

D301426X012

February 2023

For customer service and technical support, visit [Emerson.com/SupportNet](https://emerson.com/SupportNet).

North America and Latin America:

Emerson Automation Solutions
Energy and Transportation Solutions
6005 Rogerdale Road
Houston, TX 77072 U.S.A.
T +1 281 879 2699 | F +1 281 988 4445
[Emerson.com/SCADAforEnergy](https://emerson.com/SCADAforEnergy)

United Kingdom:

Emerson Automation Solutions
Meridian East
Meridian Business Park 7
Leicester LE19 1UX UK
T +44 0 870 240 1978
F +44 0 870 240 4389

Europe:

Emerson S.R.L
Regulatory Compliance Shared Services
Department
Company No. J12/88/2006
Emerson 4 Street
Parcul Industrial Tetarom 11
Romania
T +40 374 132 000

Middle East/Africa:

Emerson Automation Solutions
Energy and Transportation Solutions
Emerson FZE
P.O. Box 17033
Jebel Ali Free Zone – South 2
Dubai U.A.E.
T +971 4 8118100 | F +971 4 8865465

Asia-Pacific:

Emerson Automation Solutions
Energy and Transportation Solutions
1 Pandan Crescent
Singapore 128461
T +65 6777 8211 | F +65 6777 0947

© 2003-2023 Bristol Inc., an affiliate of Emerson Electric Co. All rights reserved.

This publication is for informational purposes only. While every effort has been made to ensure accuracy, this publication shall not be read to include any warranty or guarantee, express or implied, including as regards the products or services described or their use or applicability. Bristol Inc. (hereinafter “Energy and Transportation Solutions” or ETS) reserves the right to modify or improve the designs or specifications of its products at any time without notice. All sales are governed by ETS terms and conditions which are available upon request. ETS accepts no responsibility for proper selection, use or maintenance of any product, which remains solely with the purchaser and/or end-user. Emerson Automation Solutions, Emerson, and the Emerson logo are trademarks and service marks of Emerson Electric Co. All other marks are the property of their respective owners.