

ScadaWorks Technical Reference Manual

Copyright Notice

Copyright © 2012 Industrial Control Links, Inc. All rights reserved

The software contains proprietary information of Industrial Control Links, Inc (ICL); it is provided under a license agreement containing restrictions on use and is also protected by copyright law. Reverse engineering of the software is prohibited.

Due to continued product development this information may change without notice. If you find any problems in the documentation, please report them to us in writing. ICL does not warrant that this document is error-free.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior written permission of ICL.

Industrial Control Link, Inc.

12840 Earhart Ave.

Auburn, CA

USA

(530) 888-1800

Internet Support E-Mail: support@iclinks.com

Internet Sales E-Mail: info@iclinks.com

Website: www.iclinks.com <http://www.iclinks.com>

Contents

Copyright Notice	2
------------------	---

Getting To Know ScadaWorks	15
----------------------------	----

Typographical Conventions	17
Before you get started	18
HELP is always there	18

ScadaWorks Development Cycle	21
------------------------------	----

Getting Started with a Project	23
Creating a Project	23
Creating a Node	24
ISaGRAF And ScadaWorks	27
ISaGRAF 3 Program Environment	29
ISaGRAF 5 Programming Environment	29
Node Configuration - Downloading	30
Project Settings Reference	32
General Tab	32
Port Tab	33
Ethernet Tab	33
Node Settings Reference	34
Node Settings - General Tab	34
Target Configuration Tab	37
Node Settings - Advanced Tab	48
Routing Button	50
Routing Editor - Default Network Port	52
Routing Editor - Entry Network Port	52
Routing Editor - Entry Destination Type	52
Routing Editor - Entry Destination	52
Routing Editor - Internal Routing Entries	52
NetPorts Button	52
NetSessions Button	52

The ScadaBuilder User Interface	53
---------------------------------	----

File Menu	55
PROJECT Menu	55
NODE Menu	56
TARGET Menu	58
SETUP Menu	63
ISaGRAF Menu	64
TOOLS Menu	65

The ScadaBuilder Hierarchy	73
----------------------------	----

Using Log Files -- Data and Alarm Logging	77
---	----

Pinnacle and Later Controllers	78
Creating a Log File	80
Log Files Reference	82
Log File Setup Tab	83
XML example:	85
HTML example:	86

Read Only (Setup Tab)	87
Log File Archive Tab	87
Using Log Events	91
Creating a Log Event	92
Define your Trigger or Triggers	93
Log Events Reference	94
Using Log Alarms	97
Creating a Log Alarm	97
Log Alarms Reference	99

Registers 103

Boolean, Integers and Real Register Windows	104
Managing and Editing Registers	106
Retained (Non-Volatile) Registers and Initial Values	113
Using Registers in Your Program	114
ISaGRAF 3 Register Tools	116
Registers Reference	118
Type	119
Block Size	119
Prefix	119
Prefix Enum	119
Template File	120
Middle Enum	121
Suffix	121
Suffix Enum	121
Sample Output	121
Map to NVRAM (Make Retained)	121
Register Name	121
Index	121
Count	122
Messages Size	122
Retained Check Box	122
Comments	122
Add Button	122
File Button	123
Register List Button	123
File Export Button	123
Name List	123
I/O Checkbox	124
TUI Checkbox	124
Comm Stats Checkbox	124
Network Events Checkbox	124
Remote Scale Checkbox	124
Retained Register Drop Down List	124
Symbol Column	125
Format Button	125
Close Button	126
Show Attribute Columns	126
Buffer Format Editor	126

Using I/O Channels and Mapping Registers 129

Mapping and Unmapping I/O to Registers	131
I/O Map Reference	134
Applying a Scaling Record	135
I/O Scale Assignment Reference	136
I/O Options	136
I/O Configuration - ICL-4300 (PC-in-a-Brick) Controllers	138
I/O Scaling	139
Register Min & Max	140
Clamp Min & Max	140

I/O Mode	140
I/O Range Low & High.....	141
I/O Ranges for Different Scaling Modes	142

Using Triggers 143

Defining a Trigger	144
Setpoint Triggers	145
Alarm Suitable Triggers.....	147
Special Trigger Types	147
Triggers Reference.....	148
Type.....	148
Cycle Count.....	149
Timer Period	149
Timer Mode	149
Start Register	150
Register Block Size	150
State.....	150
Threshold Constant	150
Threshold Register	150
Buffer	151
Network Session	151
Network Address	151
Real Time Clock (RTC).....	151
Hour/Min Constants.....	151
Hour/Minute Registers.....	151
Day Checkboxes	151
Trigger - Options Tab	151

Using Local Events 155

Creating A Local Event.....	155
Defining A Local Event	155
Local Events Reference	157
Trigger	157
Add Button	157
Start Register	157
Value	157
Block Size.....	157

Using Alarms 159

Creating An Alarm.....	160
Defining An Alarm	160
Alarm Options	161
Alarms Reference	163
Activation Trigger.....	163
Acknowledge Trigger	163
Add Button	163
Option Trigger on Startup.....	163
Alarm Text.....	164
State Map.....	164
Alarm Group.....	164
Alarm Auto Log Enable.....	164

Using Local Alarms 165

Creating a Local Alarm.....	165
Defining a Local Alarm	166
Local Alarms Reference.....	167
Alarm.....	167

Alarm Annunciation Register	167
Unack Mode	167
On/Off Time	167
Ack Mode	168
Idle Mode	168

Using Alarm Dialers 169

Creating an Alarm Dialer	169
Setting Up a Call Group	171
Finishing the Alarm Dialer Configuration	173
Alarm Dialer Reference	174
Call Group	174
New Button	174
Edit Button	174
Dialer Activation	174
Alarm Message	174
Register	174
Decimal Places	174
Register Button	174
Date Button	175
Time Button	175
Preview Button	175
Settings Button	175
Call Group Reference	175
Setup Tab	175
<i>Buffer or Message Register</i>	177
Dialing Tab	177
Generating and Downloading a Voice File	179
Customizing the Voice Interface (English and Non-English)	180
Generate Voice Reference	182

Using the Voice User Interface 183

Creating a Voice User Interface	183
Setup Tab Reference	185
Controlling VUI Access	188
Login Tab Reference	188
Misc Tab Reference	189

Communications 191

Hardware, How, What and When	191
Network Port (Hardware)	192
Network Session (How The Hardware Network Port Is Used)	192
Network Destinations (Where The Data Will Be Transmitted To Or Retrieved From)	193
Network Events (What Data Is Transmitted Over The Network Session)	193
Triggers (When Will the Data Be Sent or Read)	193
Master Protocol List	195
Slave Protocol List	197
Using Network Ports	199
Network Ports Reference	199
Using a Dialup Network Port	203
Uses for Dialup Modems	204
Dialup Modem Parameters Reference	205
Using TCP/IP Over Serial and Dialup	207
TCP/IP Port Parameters Reference	208
Using a Cellular Modem Network Port	211
Signal Strength Map	212
Signal Poll Period (Sec)	212
RTC Sync Enable	212
RTC Sync Period (hrs)	212

Cell Modem Status Map	212
Cell Phone Number Map.....	213
Last Caller Number Map.....	213
Incoming Message Count Map	213
Outgoing Message Count Map.....	213
Text Message Log Enable.....	213
Using Network Sessions	214
Data Presentation	214
Creating a Network session.....	215
Protocol Specific Configuration Buttons.....	216
Network Sessions Reference	219
Using Network Destinations	241
Network Destination Reference	245
Modbus Protocol.....	249
Creating a Modbus Master Interface	249
Creating a Modbus Master Destination	252
Creating Modbus Master Events	254
Network Events Reference.....	255
Network Event Activation.....	260
Creating a Modbus Slave Interface	262
Network Event and Network Message Link Display	263
Real Time Clock Network Events	267
Understanding Modbus Routing (Store and Forward).....	269
Setting Up a Modbus Routing System	270
Using Network Message Links	272
File Transferring Over Modbus And Bricknet.....	278
Remote Host File Transfers	281
Secure Data Exchange (SDX and STM) Protocols.....	288
Creating An SDX Session.....	289
Creating An SDX Destination	291
Creating SDX Network Events	293
Understanding SDX Routing	296
Setting Up An SDX Routing Example	298
Setting Up An STM Interface	301
DF1 Protocol.....	302
Creating a DF1 Master Interface	302
Creating a DF1 Master Destination.....	303
Creating DF1 Master Events	305
Creating a DF1 Slave Interface	308
DNP 3 Protocol.....	310
Special DNP 3 Diagnostic Statistics.....	310
Creating a DNP 3 Slave Interface	312
Creating a DNP 3 Master Interface	315
Creating a DNP 3 Master Destination	316
Creating DNP 3 Master Objects.....	319
Creating DNP 3 Master Events	320
Creating DNP 3 Master Class Events	326
Supported DNP 3 Master Commands	328
Slave Configuration To Retrieve Event Data	329
Getting Event Data With a SCADA System.....	330
Ethernet IP Protocol.....	332
Creating An Ethernet IP Master Interface	332
Creating An Ethernet IP Master Destination	333
Creating Ethernet IP Master Network Events.....	336
Creating An Ethernet IP Slave Interface	339
Creating Ethernet IP Network Message Links.....	339
Hart Master Protocol.....	341
Creating a Hart Master Interface	341
Creating a Hart Master Destination.....	342
Creating Hart Master Events	343
Command (Hart Supported).....	345
Bricknet Protocol.....	348
Creating a Bricknet Interface	348

Creating a Bricknet Destination	350
Creating Bricknet Network Events.....	352
Understanding Bricknet Routing.....	355
Setting Up a Bricknet Routing System.....	355
File Transferring Over Modbus And Bricknet.....	359
Using Bricknet To Communicate With I/O Modules	361
Numeric and Alpha Numeric Pager Sessions	366
Email Protocol.....	367
Creating an Email Interface	368
Creating Emails	370
Email Reference.....	372
Creating an FTP Client Interface.....	374
On the Login Tab of the Network Session	376
Creating an FTP Event.....	377
File Configuration.....	378
FTP Event Reference	379
Using Remote Scaling.....	381
Creating a New Remote Scale Record	382
Scaling a Remote Device's Register	383

Textual User Interface (TUI) 385

Creating a TUI.....	389
TUI Menu ItemsTUI Designer Menus	390
TUI Menu	390
Groups	390
Group Menu.....	391
Page Menu	391
Tools and Editing.....	391
Rubber Band.....	392
The Right Click.....	392
General Settings.....	392
Name	393
Port	393
Port (2nd).....	393
Mode	393
Header	394
Columns.....	394
Rows.....	394
Refresh	394
Scan Rate.....	394
Auto Edit	394
Quit Enable.....	395
Telnet Max.....	395
Telnet Port	395
Security Register.....	395
Telnet Timeout	395
Scheme	395
Security Level.....	395
Attributes	396
Header Attributes.....	396
Text Attributes.....	396
Link Attributes.....	396
Selection Attributes.....	396
Value Attributes.....	397
Graph Attributes	397
Button Attributes	397
Disable Attributes	397
Page Settings.....	397
Name	397
Prompt	397
Group.....	397

Security Level	398
Attributes	398
Text Attributes	399
Value Attributes.....	399
Button Attributes	399
Page Link Attributes	399
Controls	399
Control Parameters	400
TUI - Alarm Settings	401
TUI - Bar Graph Settings	401
TUI - Buffer Field Settings	403
TUI - Button Settings.....	404
TUI - Log Settings	405
TUI - Page Link Settings	406
TUI - Register Field Settings	407
TUI - Text Settings	408
TUI - Time and Date Settings	409
TUI - Value List Settings	410
TUI Designer - Label List Editor	411
TUI Designer - Value List Editor.....	412
TUI Local HMI (Pinnacle Controllers)	415
Local HMI Navigation	416
Creating A Local HMI	416
Local HMI Numeric Entry	421
User Portal (Web Interface)	423
Setting Up an Administrator Account	424
Accessing the User Portal for Administration	426
Adding a New User Account	427
User Portal Permissions	428
Editing and Deleting User Accounts	430
Adding and Editing Web/HMI Links	432
Adding and Accessing Web/HMI Links Via a User Account.....	434
Changing The IP Settings	436
Setting Radio Diagnostic Mode	438
Synchronizing The Real Time Clock	440
Changing Your Password.....	441
Live View -- View and Edit Registers.....	442
Boolean Registers	444
Integer and Real Registers	445
Message Registers.....	446
Trend Graphs	447
Alarms	448
Generating Alarm and Alarm Group Reports	449
Accessing Alarms in Alarm Groups.....	450
Active Alarms! Popup.....	452
Documents	453
Trending	455
Setting Up a New Trend.....	456
Trend Parameters	458
Configuring Traces and Trace Parameters.....	463

Accessing a Trend Web Page	467
Reports Button.....	468
Traces Button.....	471
Settings Button	472
Date Range Button.....	475
Info Button.....	477
Stats Button	478
Refresh Button.....	478
Mode (Strip/History Selector).....	479
Zooming (PC Browsers)	480
Cursor: Mode Hovering and Zooming With iPad and iPhone Browsers.....	481

Text Message Interface (TMI) 483

Logging In To The TMI.....	485
Using More Complex TMI Grouping.....	485
Text Message Interface (TMI) Command Reference	488
Text Message Interface (TMI) Reference.....	490
General Tab	490
Registers Tab	491
Alarms Tab	491
Users Tab.....	492

Mappings Reference 493

Reboot Map	494
Status LED Map	494
Memory Available Map.....	494
Clock Speed Set / Map.....	494
Device To Register Mapping	494
Time Zone Map/Constant	495
Daylight Savings Map.....	495
Controller Serial Number Map.....	495
FTP Server Status Buffer	496
FTP Client Status Buffer.....	496
HTTP Server Status Buffer	496

Global Positioning Satellite (GPS) Interface 497

GPS Reference.....	497
Network Port.....	498
RTC Sync Period	498
Comm Error Register.....	498
Comm Timeout.....	498
Task Period	498
Data Mapping	498
Delta Mapping	499
Required GPS Messages	499

ISaGRAF Function and Function Blocks 501

ISaGRAF Data Types and Function Prototypes	501
Low Level Communications Functions.....	504
Open a Communications Port -- ComOpen()	505
Close a Communications Port -- ComClose()	506
Reset Communication Port Parameters -- ComSet().....	507
Get The Number of Bytes In The Receive Buffer -- ComRCnt()	508
Read A Byte Out Of The Receive Buffer -- ComRdBt()	509
Read Multiple Bytes Out Of The Receive Buffer -- ComRdBuf().....	509
Write A Byte To The Transmit Buffer -- ComWrBt()	510
Write Multiple Bytes To The Transmit Buffer -- ComWrBts()	510

Write A Message To The Transmit Buffer -- ComWrStr()	512
Detect When The Transmit Buffer Is Empty -- ComXmtEm()	513
Clear The Receive Buffer -- ComClrRcv()	514
Control RTS On The ComPort -- ComRts()	515
Control DTR On The ComPort -- ComDtr()	516
Read CTS From The ComPort -- ComCts()	517
Read DCD From The ComPort -- ComDcd()	518
Open A Network Port By Name -- NPOpen()	519
Get A Netport Handle By Name -- NPHandle()	521
Send A Packet Over A Network Port -- NPPktSnd()	523
Close A Network Port By Handle -- NPClose()	525
File I/O Functions	526
Open a File For Appending -- f_aopen()	526
Open A Binary File For Read Only Access -- f_ropen()	527
Open A Binary File In Read-Write Mode -- f_wopen()	528
Check For End Of File Status -- f_eof()	529
Write An Integer To A File -- fa_write()	530
Write A Message Register to a Line in a Text File -- fm_writecrlf()	531
Write A Message Register (STRING) To A File -- fm_write()	533
Write A Real Register To A File -- fr_write	534
Seek Or Get The Position Of A File -- f_seek()	535
Read An Integer Register From A File -- fa_read()	536
Read A Line From a Text File to a Message Register -- fm_readcrlf()	537
Read A Message Register (STRING) From A File -- fm_read()	539
Read A Real Value From A File -- fr_read();	540
Close A File -- f_close()	541
Check To See If A File Exists -- f_exist()	542
Delete A File -- f_delete()	543
Rename A File -- f_rename()	545
Copy A File -- f_copy()	546
Copy A Large File -- f_copy_l()	548
Check Disk Space Or Create A Directory -- diskmgt()	552
Network Control Functions	554
Enable Or Disable A Network Session By Name -- NSCtrl()	554
Retrieve NetEvent, NetDest, NetSession Handles -- nethandles()	555
Retrieve A Network Session Handle -- nshandle()	557
Retrieve A Network Destination Handle -- ndhandle()	559
Retrieve A Network Event Handle -- nehandle()	560
Manually Trigger A Network Event -- nettrigger()	561
Read Network Event State -- nepending()	563
File Transfer Protocol (FTP) Functions	565
Open An FTP Connection -- FtpOpen()	565
Close An FTP Connection -- FtpClose()	565
Get A File Over FTP -- FtpGet()	566
Send A File Over FTP -- FtpSend()	566
Get The FTP Client Status -- FtpCStat()	567
Change FTP Transfer Type -- FtpType()	567
Low Level I/O Port Access Functions	569
Read The Eight Position DIP Switch Value -- Read_sw()	569
Read One Byte From I/O Space -- InByte()	569
Read A 16bit Word From I/O Space -- InWord()	570
Write A Byte Out To I/O Space -- OutByte()	570
Write A Word Out To I/O Space -- OutWord()	571
Bit Packing and Unpacking functions	572
Pack 16 Booleans Into An Integer Register -- pack16()	572
Unpack 16 Booleans From An Integer Register -- Unpack16()	573
Instrumentation Functions	575
PID Closed Loop Control -- ICLPID	575
Apply Scaling Record To Analog Channel -- ioscale()	580
Characterize A Non-Linear Instrumentation Curve -- charctrz()	581
Enable Or Disable The Controller I/O Scan -- IOCtrl()	583
Scale A Linear Analog Device -- Scaler()	583
Totalize The Time A Boolean Is True -- Runtime()	584

Periodically Totalize An Analog Value -- Totalize()	584
Track And Hold An Analog Control Value -- trackhld()	585
Limit Rise And Fall Rate Of An Analog Value -- ratelim()	586
PID Closed Loop Control -- PID_AL()	587
Variable Access Functions	591
Read An Integer Register By Index -- AnaRd()	591
Write To An Integer Register By Index -- AnaWr()	591
Read A Real Register By Index -- RealRd()	592
Write To A Real Register By Index -- RealWr()	592
Read A Boolean Register By Index -- BooRd()	592
Write To A Boolean Register By Index -- BooWr()	593
Read A Message Register By Index -- MsgRd()	593
Write To A Message Register By Index -- MsgWr()	594
Logical Functions	595
Flip Flop Latch Function -- flipflop()	595
Logging Functions	596
Use The Real Register Dictionary As A FIFO Log Or Trend -- Logreal()	596
Use The Integer Register Dictionary As A FIFO Log Or Trend -- Logana()	598
Real Time Clock (RTC) Functions	600
Write The Time To The RTC From Integers -- Timewr()	600
Write The Date And Time To The RTC From Integers -- Datewr()	600
Write The Current RTC Seconds Since 00:00 01/01/70 -- RTCSecWr()	601
Read The RTC Into Integers -- DateRd()	602
Read The Current RTC Seconds Since 00:00 01/01/70 -- RTCSecs	603
Read The Time And Date From GPS -- gpsrd()	603
Read Date, Time Or Day Of Week -- day_time	604
Get The Maximum Scan Time -- scanmax()	605
Get The Scan Time Of The Previous Scan -- scantime()	605
Get The Number Of Milliseconds Since Startup -- systick()	606
Redundancy Function Block For Legacy Controllers	607
Redundnt Function Block	607
Technical Considerations	608
Program Implementation	609
Deployment	612
General Notes	612
Redundancy Function Block for Pinnacle Controllers	614
Setup And Technical Considerations	614
Redundant Function Block Parameters	616
Program Implementation	623
Deployment	627
ISaGRAF Socket Functions (UDP and TCP/IP)	629
Client Connection Basic Strategy	629
Server Implementation Basic strategy	629
Allocate a new socket -- SockAlloc()	629
Connect To a Server or Initialize Socket -- SockConnect()	630
Open a Listening Socket To Act As Server -- SockListen()	631
Write Data To a Socket -- SockWrite()	632
Read Data From a Socket -- SockRead()	635
Release a Socket -- SockRelease()	636
Free A Socket From Memory -- SockFree()	637

ISaGRAF HiBeam Web HMI **639**

Getting Started	639
HiBeam Registration.....	639
Starting a HiBeam Project.....	640
Setting Up Accounts	641
Setting Up a HiBeam Screen Builder Project	642
Building Our First Screen	645
Downloading Our Screens	646
Accessing Our New Page From a Web Browser	648

Appendix A, An Ethernet/Internet Primer for TCP/IP **649**

TCP/IP Addressing	649
Private Networks	650
Routing	650

Index **655**

Getting To Know ScadaWorks

ScadaWorks is a complete Windows development environment for Supervisory Control And Data Acquisition (SCADA) systems. With ScadaWorks, everything from simple alarm dialers to large multi-controller systems can be created with a graphical “fill in the blanks” environment and little or no programming.

ScadaWorks consists of two main components:

ScadaBuilder - Used to create all of the SCADA system functions that are not programmable logic, including:

- Serial and Modem Communications
- Ethernet Communications
- Internet Services: FTP file transfer, E-mail, and serving web pages
- Alarm Annunciation, from light panels to voice messages and paging
- Data and Alarm Logging
- Voice User Interface (touchtone remote control)
- Text User Interface (simple HMI interface over serial link or Ethernet)
- GPS (Global Positioning Satellite) time and position updates

ISaGRAF - An IEC 61131-3 standard logic and math programming environment that supports six different languages including:

- Ladder Logic
- Structured Text
- Function Block Diagram
- Flow Chart
- Sequential Function Chart
- Instruction List

ScadaWorks is designed to minimize the time, effort and cost of creating SCADA systems of all types.

ISaGRAF IEC 61131-3 Programmable Logic

ScadaWorks includes the ISaGRAF IEC 61131-3 programming software from ICS Triplex. ISaGRAF is the most popular implementation of the IEC 61131-3 standard with the largest number of installed systems worldwide. ISaGRAF includes a full development environment for editing, downloading, debugging and documenting control programs. The variable (tag) database, download and archival tools are shared with all of the other ScadaWorks components, eliminating duplicate entry work and providing a fully integrated SCADA system development tool.

Serial and Modem Communications

ScadaWorks simplifies serial communications over a variety of media including UHF, VHF and Spread Spectrum radio as well as dial-up/leased-line telephone lines, fiber optic cable, RS-232 links, and RS-485 networks. Any controller serial port can have one or more standard protocols assigned to it including:

- Modbus (RTU/ASCII/TCP/UDP, Master/Slave)
- DF1 (Allen Bradley Master/Slave)
- DNP 3
- Hart (Master)
- Bricknet (ICL Peer-to-Peer)
- Alphanumeric and Numeric Pager
- FTP (Client)
- E-mail (Client-Outgoing)

If the serial port has a dial-up type modem, ScadaWorks automatically dials and establishes a link when needed, as well as accepting incoming calls.

A serial port may be configured to serve as an ISaGRAF and ScadaBuilder download and debugging port using ScadaWorks. New controller configurations and logic programs can be transferred over hard-wired, radio, or telephone serial links. No programming beyond fill-in-the-blanks configuration is required for any of the standard serial communications features, so creating a communications network is quick and painless.

High-speed Ethernet Communications

Most ICL Controllers have a built-in Ethernet port. ScadaWorks is used to set the functionality of this port, including ISaGRAF and ScadaBuilder program downloading and debugging, Modbus TCP (Master or Slave) register access, incoming or outgoing file transfers to the Controllers flash disk using FTP protocol, outgoing e-mail with optional file attachments, and serving HTML pages using HTTP protocol.

Internet Services

ICL Controllers can take full advantage of the Internet; accessed over local area networks, DSL lines or dialup connections. Programs, data files and log files can be passed back and forth using FTP protocol, register data can be examined and changed using Modbus TCP, reports and alarms can be sent out as e-mails, and low-cost graphical web based user interface programs such as ErgoView can utilize the HTTP web server capabilities.

Alarm Annunciation

ScadaWorks has a complete alarm annunciation package. Alarms can be configured to control indicator lights in standard ANSI sequences, dial out and contact another computer, dial out and vocalize spoken messages and/or send messages to cell phones and pagers. The alarm handling facilities support both local and remote alarm acknowledgment.

Data and Alarm Logging

ICL Controllers come with either 4, 8 or 16MB of flash memory configured as a “disk drive”. A portion of this memory (approximately 1 MB typically) is used to store programs and configuration files, but the majority of the memory is available as storage for data log and alarm log files. The Controller can manage multiple log files simultaneously.

The format of the log files can be directly compatible with common spreadsheet and database programs, or compressed to optimize the use of the available storage space, and uncompressed by a separate PC based program. The log files can be accessed with standard communications tools such as FTP file transfer programs while the controller is operating and executing a logic program. A LogGrabber application for PCs is also available to automate collection of log files on a schedule from one or multiple controllers.

Voice User Interface (VUI)

ICL Controllers, with an available Voice/Modem option, can be configured with ScadaWorks to allow access to register values and sensor readings via conventional telephone. The values in registers are vocalized by synthesized voice. A touchtone keypad can be used to change setpoints and control parameters. Password control is configurable to protect against unauthorized access.

Text User Interface (TUI)

ScadaWorks provides a simple Text User Interface (TUI) for applications that don't require or don't support a full blown graphical interface. A typical example is the 4 line x 20 character Viewpoint Operator Interface Terminal. The

TUI can also be used with PCs over a serial or Ethernet link. The TUI is ideal for use over slow communications links and as diagnostic tools that can be set up in seconds without the overhead of a graphical environment.

GPS

ICL Controllers can accept GPS time and position update information to provide fraction-of-a-second accuracy for data logging and time critical control functions as well as mobile applications requiring accurate position data. Generally, the GPS function is used to enhance the performance of the Real Time Clock built into every ICL controller.

ScadaWorks Minimum System Requirements

- Operating system: Windows 98SE-(minimum) Windows XP SP2-(recommended)
- 64 MB of RAM-(minimum) 128 MB RAM-(recommended)
- 50 MB of available hard disk space.
- Microsoft Internet Explorer 6.0 or later.
- Internet access with at least a 28.8 Kbps connection (For online updates)
- .NET Framework 1.0 (Included during install)

In This Section

Typographical Conventions	17
Before you get started.....	18

Typographical Conventions

Before you start using this guide, it is important to understand the terms and typographical conventions used in the documentation. For more information on specialized terms used in the documentation, see the Glossary at the end of this document. The following kinds of formatting in the text identify special information.

Formatting convention*Emphasis*

CAPITALS

KEY+KEY

Menu | Menu Item**Type of Information**

Use to emphasize the importance of a point or for variable expressions such as parameters.

Names of keys on the keyboard. for example, SHIFT, CTRL, or ALT.

Key combinations for which the user must press and hold down one key and then press another, for example, CTRL+P, or ALT+F4.

The | (pipe) symbol is used to show menu and tab selections. For example, "**Tools | File Transfer...** menu" means to click on the Tools menu and select the File Transfer... menu option.

When discussing a configuration dialog, this will define where the dialog exists and how to get to it.

For example, if a parameter is found at "**Node | Settings | Ethernet / Serial IP** tab", you can navigate to that parameter by clicking the Node menu and selecting the menu option Settings, then clicking on the Ethernet / Serial IP tab.



This kind of box means it contains information that could be very useful to you.



This kind of box means that this is something to watch out for or has been a support issue in the past, be careful!

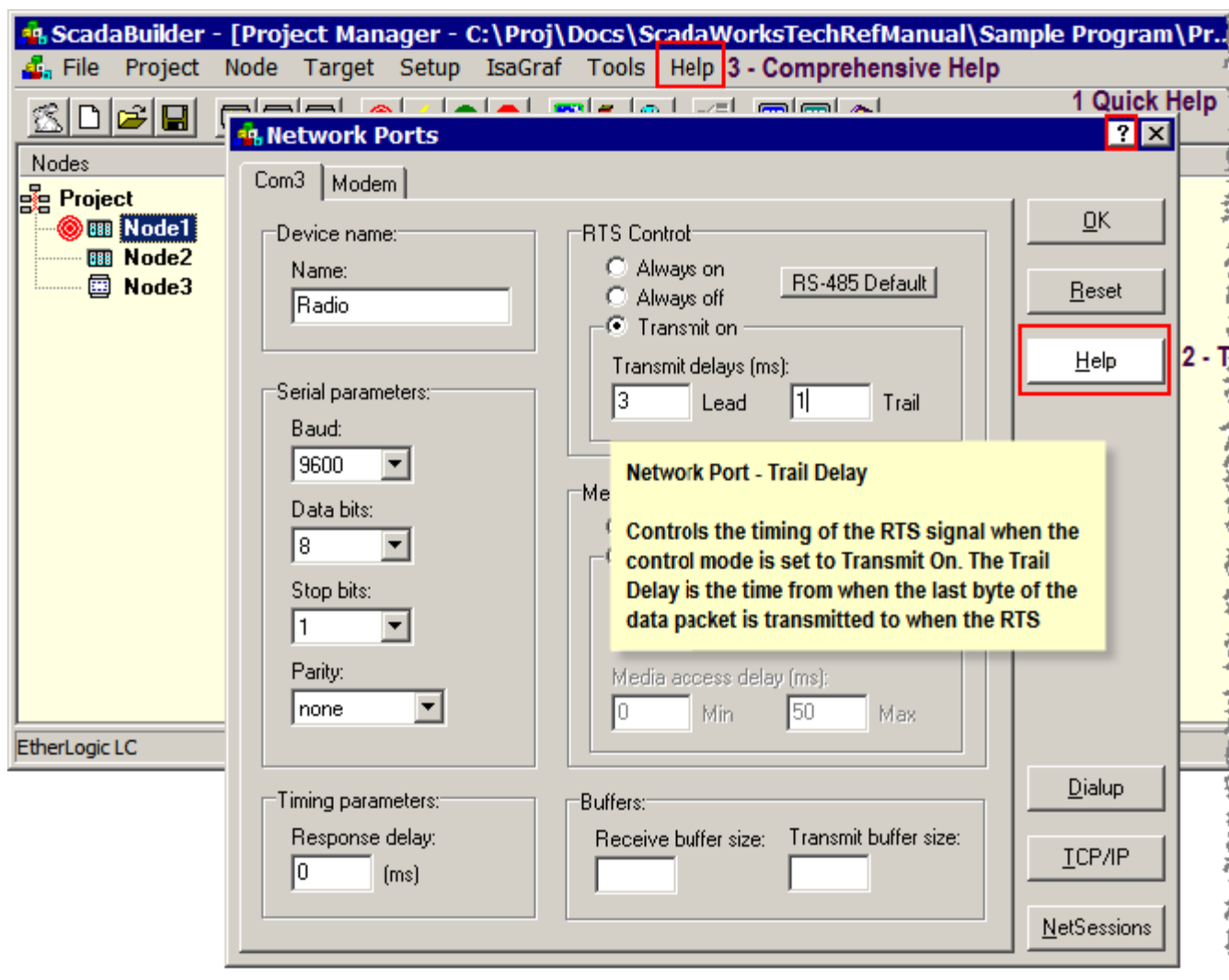
Before you get started

HELP is always there

ScadaWorks is a very large program. If you're a novice ScadaWorks user, you're going to have a few questions. Luckily, there's great on-line help that gains you access to this very guide:

1. Quick Help

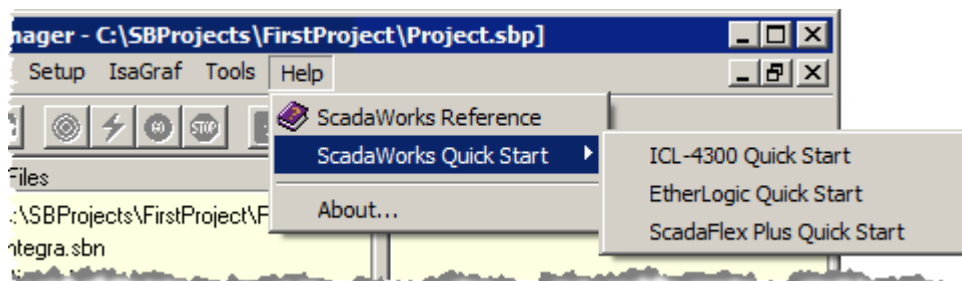
The fastest way to get help on a specific item is to click on the "?" button in the upper right-hand corner of every window, then click on the item itself.



2. Topic Help

Want help on the entire Window? Click on the Help button in the selected Window and ScadaWorks will display help for all of the fields in that Window.

3. Comprehensive Help



Need more help? Then select the “Help” menu item on the Main Window for access to tutorials, reference manuals, and an on-line copy of this manual! The ISaGRAF section of ScadaWorks has a similar button with detailed programming help. Also, don’t forget your controller’s Quick Start Guide if you’re just getting started.

ScadaWorks Development Cycle

ScadaWorks is a tool for developing entire SCADA projects.

A SCADA project consists of one or more NODES. A node is a controller or RTU, of which there are typically one at each site in the project.

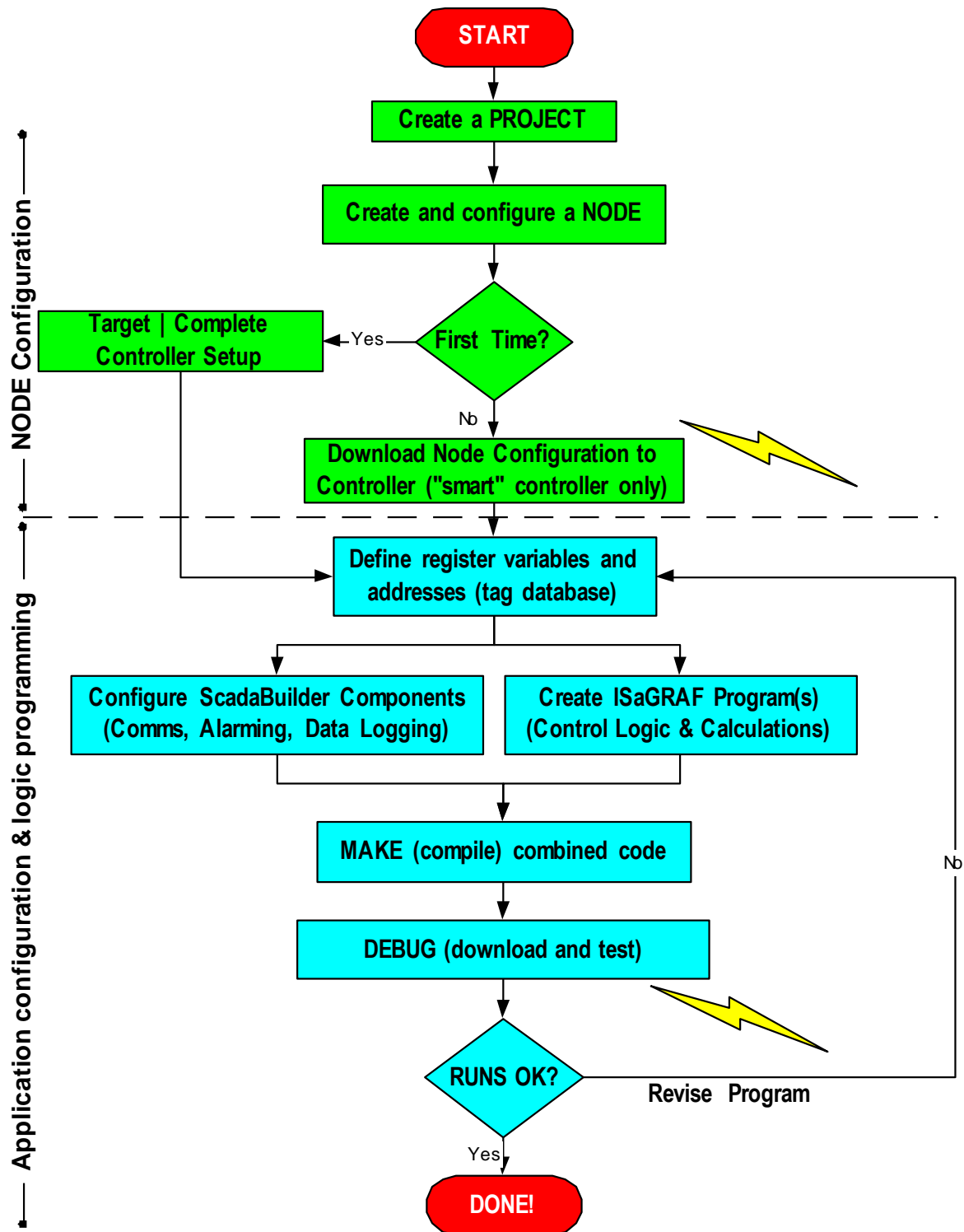
“Smart” controller nodes (those that are programmable) must have at least a minimum amount of configuration to define how ScadaWorks will communicate with the node for downloading and debugging. Furthermore, if the node is going to be connected to the Internet or an Ethernet network, the node must have additional configuration information defined including addressing, user ID and security access codes. All of these items are typically defined and downloaded when the node is first set up, although they can be changed and updated at any time.

Once a node configuration has been created and downloaded, the application specific configuration and logic programming can be created.

The two main components of ScadaWorks used to build SCADA applications are ISaGRAF and ScadaBuilder. These are actually two Windows applications that have been integrated together into a single software package called ScadaWorks. Both applications share a common database for variables (registers). ScadaBuilder uses these registers in its messages to share data with other controllers and computers, to log data to files and to process alarms. ISaGRAF manipulates (output) registers based on user programmed control strategies and calculations, and register input information such as setpoints and sensor data.

The first step in creating a Node (controller) application is to define the register database. After the registers are defined (or later added), ScadaWorks editing tools are used to define the ScadaBuilder configurations and ISaGRAF tools are used to create logic programs.

Under ScadaWorks, the two components are compiled together at once (a process called “MAKE”) and then downloaded to the Controller’s flash disk and tested using ISaGRAF tools (“DEBUG”) over a serial or Ethernet communications link. This process is repeated as program changes are made and new code is verified.



In This Section

Getting Started with a Project	23
ISaGRAF And ScadaWorks.....	27
Node Configuration - Downloading	30
Project Settings Reference	31
Node Settings Reference.....	33
Routing Button.....	49

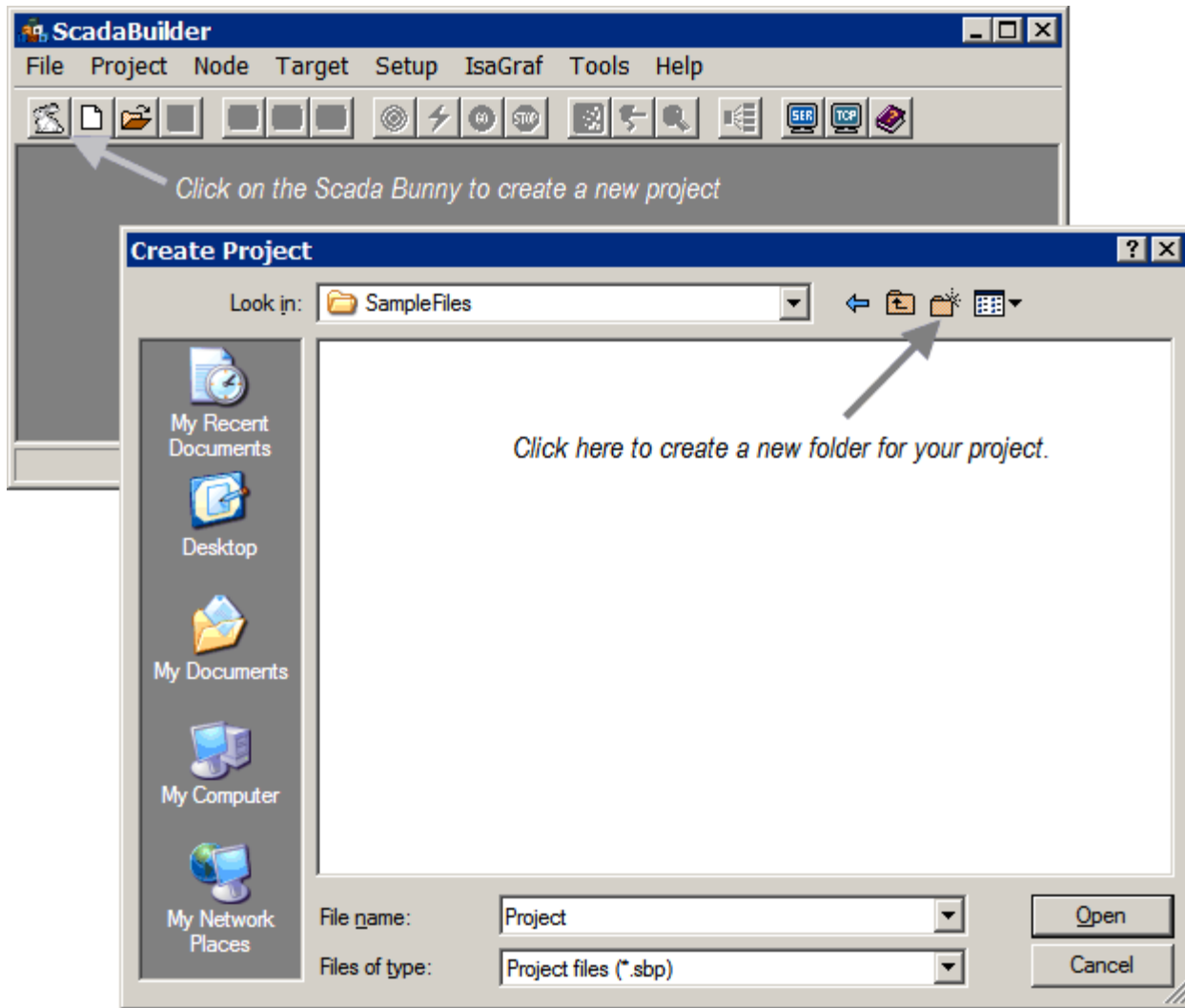
Getting Started with a Project

ScadaWorks provides a development environment that simplifies managing the design of SCADA projects. A ScadaWorks project consists of one or more controllers or RTUs called “nodes”. Each node has its own configuration and/or programming.

Creating a Project

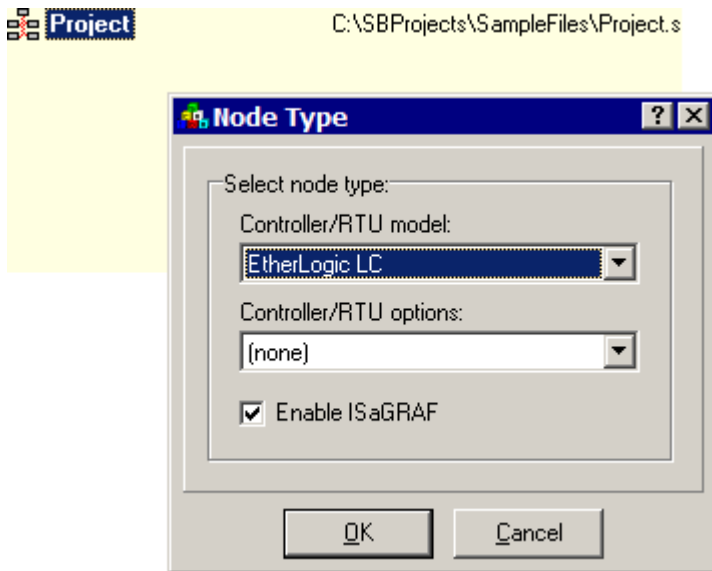
When ScadaWorks is first started, it prompts for a project folder in order to start a new project. Typically, you will need to create a new folder since ScadaWorks requires that a project have its own dedicated folder. Two ScadaWorks projects may not share a folder.

In the future, you can start a new project by clicking on **File | New Project** menu, or even easier, click on the “ScadaBunny” icon in the upper left-hand portion of the ScadaWorks main window. The ScadaBunny is a shortcut tool to start a new project quickly.



Creating a Node

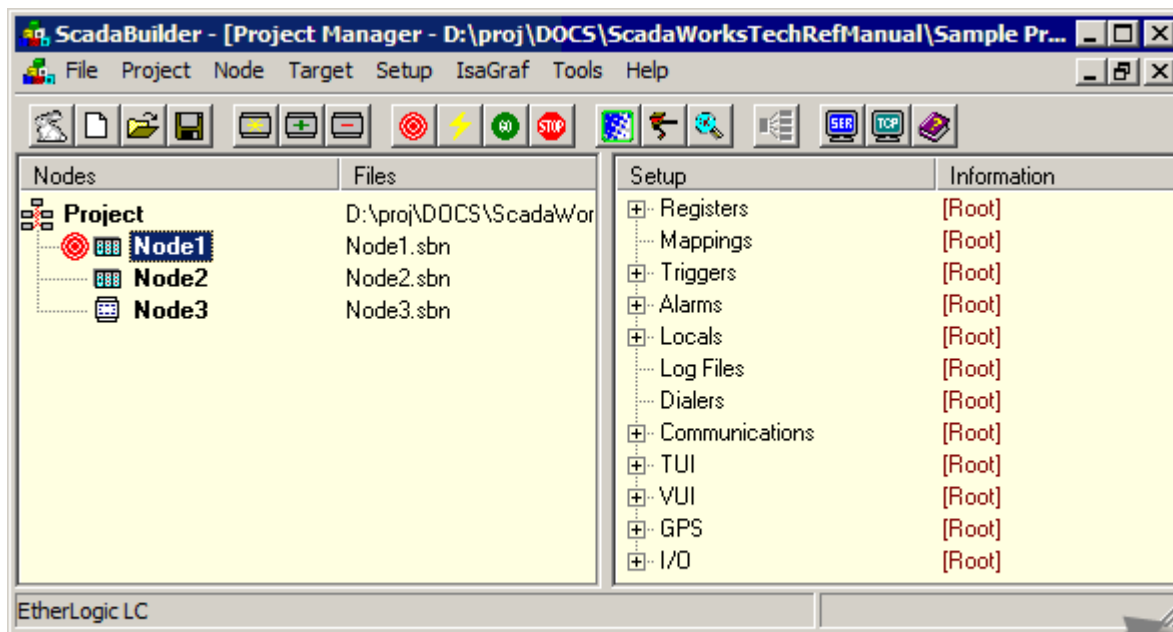
You can click on the “New Node” hot button to create a node, or you can click on the **Node | New** menu. You will be prompted for a name and configuration information needed to create a node.



When using the ScadaBunny to create a project, the first node is created automatically. Its default name is “Node1”. Whenever you manually create a new node, you will be prompted for a name. In either case, the node name (as well as the hardware type and configuration) can be edited and changed by simply double-clicking on the existing node name, or by clicking on **Node | Settings** menu and selecting the General tab.

ScadaBuilder applications can be used either by themselves or in conjunction with ISaGRAF. ISaGRAF adds local programming and logic capabilities using any combination of six standard languages including ladder logic, function blocks, flow diagrams, etc. If you don’t need any local control, ScadaBuilder can be used by itself to perform serial, wireless and Ethernet communications, data logging, alarm annunciation, e-mailing and operator interface functions. When ISaGRAF is purchased with ScadaBuilder, the entire package is known as SCADAWORKS. If you have purchased SCADAWORKS and want to use ISaGRAF, you must check the “Enable ISaGRAF” box when setting up the controller type. We recommend that you include ISaGRAF for now.

As you create nodes for your project, they are displayed in the main window of the Project Manager. Even “dumb” (non-programmable) remote I/O like ICL PicoBricks, MicroBricks MAXIO’s and ScadaFlex RTU’s can be defined in ScadaWorks so that their registers can be easily linked to “smart” controllers in the SCADA system and displayed in the project overview. The Project Manager with a typical three node (two controllers plus an RTU) SCADA system is pictured below.



Grab and Drag this corner with you mouse to resize the window as we have for these examples.

Note that the Controller that is currently being worked on is indicated by a target icon next to its name, in this case; "Node1". The type or model number of the selected Controller or RTU is displayed in the lower left hand corner; in this case, an EtherLogic LC.

The names Node1, Node2, etc. shown in the example are default names that can be changed to more descriptive labels such as "Pump Station 1" or "Tank 2" if desired. The "card rack" icon between the target and the node name indicates that this Controller is "smart". This is also evident from the selection of ScadaWorks elements available for this controller on the right hand side.

In this example, Node3 is a "dumb" RTU (MicroBrick Combo RTU) so when it's selected, only the registers are available for naming and linkage into the rest of the SCADA system. This is shown at the top of the next page.

Once you have created one or more nodes, you're ready to do some basic configuration work. You can either double-click on the node, or with a node selected (red target next to its name), click on the **Node | Settings** menu. Under the General tab, you will see the information that you originally entered when you created the node. For more detailed information on parameters in the Node Settings dialog, please refer to the *Node Settings Reference* (on page 33).



Remember to download a Node's configuration after making changes.

*All of the configuration information described in the remainder of this section affect the Nodes basic operating configuration. This information must be downloaded using the **Target | Complete Controller Setup...** menu option.*

Project D:\proj\Help\ModbusDestinations\Project.sbp
 Lassen Node1.sbn
 LC LC.sbn

Node Settings

General | Target Configuration | Advanced

Node properties:

Node name: Lassen Node address: 0 (default)

Controller/RTU model: Lassen Controller

Controller/RTU options: (none)

Regenerate default I/O scaling entries.

☐ Override project's PC port settings (ZModem):

Port: COM1 Baud: 115200

Data bits: 8 Stop bits: 1 Parity: none

Runtime options:

Trigger scan rate: 10 (ms) Trigger init delay: 0 (ms)

Counter scan rate: 100 (ms)

☐ Enable CPU watchdog:

Watch dog timeout: 10000 (ms)

☐ Enable I/O scan sync trigger:

I/O scan sync rate: 100 (ms)

Remote file hosting:

Host node: (none)

Target access options:

Internet IP address (WAN): Node file transfer mode: Prompt Serial number: 00078279

OK Cancel Help

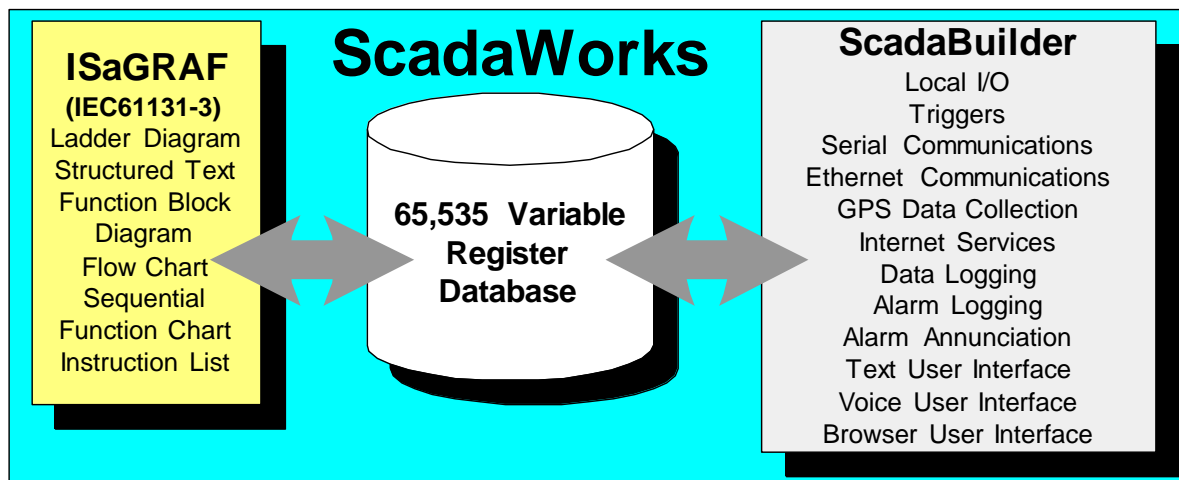
Routing NetPorts NetSessions



Caution when changing Controller Models. Once you have started programming the controller, changing the controller type may force ScadaBuilder to delete some of your work to accommodate the changes in I/O capabilities and configuration of the new controller hardware.

ISaGRAF And ScadaWorks

ISaGRAF is a software development program that conforms to the international IEC 61131-3 programming standard. ISaGRAF has been integrated with ScadaBuilder to form the ScadaWorks package and it is used for all logic, control and computation programming of ICL controllers. It operates on data in registers that are part of a database shared with ScadaBuilder.



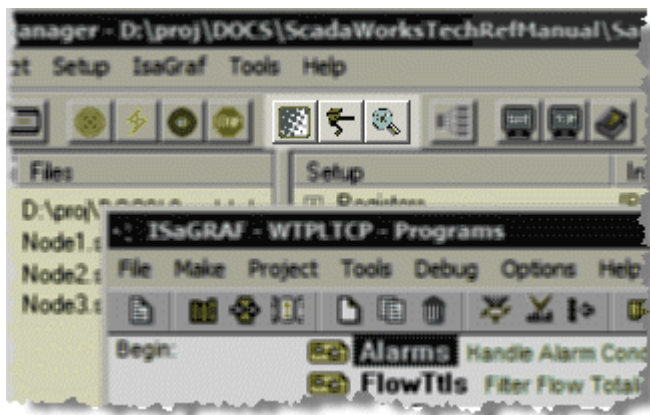
ScadaBuilder manages data communications; reading and writing to the database, logs data from the database, monitors values in the database to annunciate alarms, and displays and facilitates changes to registers in the database with human machine interfaces (HMIs) using computers, voice and touchtone telephones.


ISaGRAF, operating like a conventional logic-scanning PLC, reads in data from the database, performs logic and arithmetic calculations on the data, and writes the resulting values back to the database for use by ScadaBuilder.




Getting to ISaGRAFISaGRAF is accessed from the ScadaBuilder Project Manager through the ISaGRAF menu or with the three ISaGRAF “hot buttons”.

The ISaGRAF Workbench is used to create and edit controller programs.

Each time a change is made to a program, it must be verified for syntax errors, combined with the ScadaBuilder components, compiled into a compacted program code that executes in the controller and downloaded to take affect.



This is done with the ISaGRAF Make  function. This function is also automatically done when files have changed since the last download (or if a compile has never happened at all).

Once the ISaGRAF program is successfully compiled, it is downloaded and tested using the ISaGRAF  Debugger, the Target button  or the Lightning Bolt  button. See the **TARGET Menu** (on page 57) section for more details.

Starting the ISaGRAF WorkBench opens the main ISaGRAF window. From this window, you can actually perform all of the ISaGRAF functions; editing, “making” and debugging programs, but you’ll most likely find it easier and faster to use the ScadaBuilder “Hot Buttons” that automatically open and close Windows saving a lot of extra “mouse clicks”.



This is not intended as an ISaGRAF programming manual. For more information on ISaGRAF programming, please refer to the

ISaGRAF Workbench Manual 3.40

http://www.iclinks.com/public_ftp/DocRelease/ISaGRAFWorkbench/v3.40/ISaGRAFWorkbench3.40.pdf

For Pinnacle and later controllers, please refer to the *ISaGRAF 5.20 Manual*

http://www.iclinks.com/public_ftp/DocRelease/ISaGRAFWorkbench/v5.20/WorkbenchV5.20.pdf

ISaGRAF 3 Program Environment

Used for Etherlogic, ScadaFlex Plus and ICL-4300 Controllers.



This is not intended as an ISaGRAF programming manual. For more information on ISaGRAF programming, please refer to the

ISaGRAF Workbench Manual 3.40

http://www.iclinks.com/public_ftp/DocRelease/ISaGRAFWorkbench/v3.40/ISaGRAFWorkbench3.40.pdf

ISaGRAF 5 Programming Environment

The ISaGRAF 5 programming environment is used for all Pinnacle and later controllers.



This is not intended as an ISaGRAF programming manual.

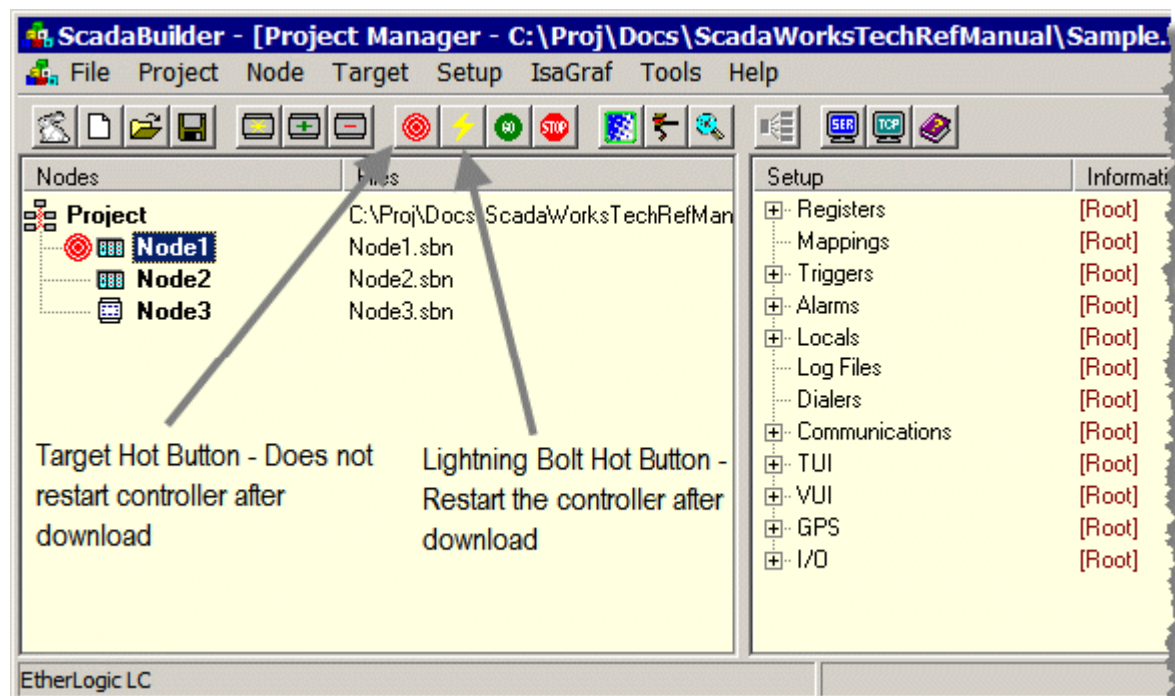
For more information on ISaGRAF programming for Pinnacle and later controllers, please refer to the *ISaGRAF 5.20 Manual*

http://www.iclinks.com/public_ftp/DocRelease/ISaGRAFWorkbench/v5.20/WorkbenchV5.20.pdf.

Node Configuration - Downloading

When a node is created or changed (i.e. enabling Ethernet, sockets, HTTP, FTP or changing an IP address, password or user ID), the node configuration **MUST** be DOWNLOADED and the controller **RESTARTED** to take effect.

When downloading over a serial link, you can also use the “Target” hot button (**Target | Send Startup Config...** menu) that leaves the node stopped, or the “Lightning Bolt” hot button (**Target | Send Startup Config and Start...** menu) that automatically restarts the controller after the download. When using an Ethernet link, you can use these same tools, but the automatic restart feature is not available. To manually restart the controller, press the controller reset button or cycle power to the controller.



"Complete Controller Setup..."


When you receive a new controller from the factory, or want to start out fresh with a controller from the field, ScadaBuilder has a menu selection to set up all of the files and restore the controller to the configuration required by the current ScadaWorks program. Note that all of the functions performed under this menu selection are available as individual menu items under the “Target” menu. This is simply an automated “do it all at once” shortcut.

From the Project Manager, click on the current node and click on the **Target | Send Complete Controller Setup...** menu.

For Etherlogic, ScadaFlexPlus, and ICL4300...

This function is **ONLY** available for downloading over a direct connect serial link. You will need a null modem cable attached to COM 1 of the controller to a serial port of your controller. To assign the proper serial port on your PC, see the Project Settings | **Port Settings** (see "Port Tab" on page 33) dialog. You will be prompted to restart the controller to initiate the download.

For Pinnacle Series Controllers...

Power cycle the unit while holding the button next to the Status LED until the controller's display shows "SB Loader". and release the button. Press the discovery  button to retrieve the controller's serial number. In the display, select the controller you want by clicking on the line. If there is more than one, make sure the serial number matches the controller you want to download to.



If this does not work, try disabling all wireless adaptors (if you are using a direct link), or disabling any firewall software. Discovery will only work on a direct non-routed link that supports TCP/IP broadcast packets. You can always type the serial number manually into the Node | Settings | General Tab | Serial Number field. The serial number will be shown on the LCD display of the controller.

When prompted, select Network to download all files. To always download the controller over the network without this prompt, you can disable it by going to Node | Settings | General Tab | Node file transfer mode and changing it to "Network" instead of "Prompt".



Serial downloading is used only in case of emergency when TCP/IP policies cannot be overcome at initial download time. It requires a null modem adaptor connect to Com 1 of the Pinnacle controller. It will not download default files for the web interfaces on the controller and has no access to subdirectories on the controller's file system.

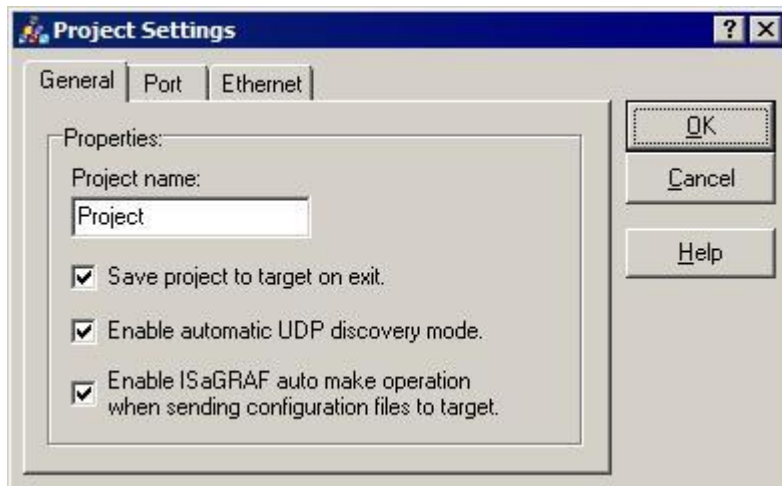
When done you will be prompted to reset the controller again to start the application.



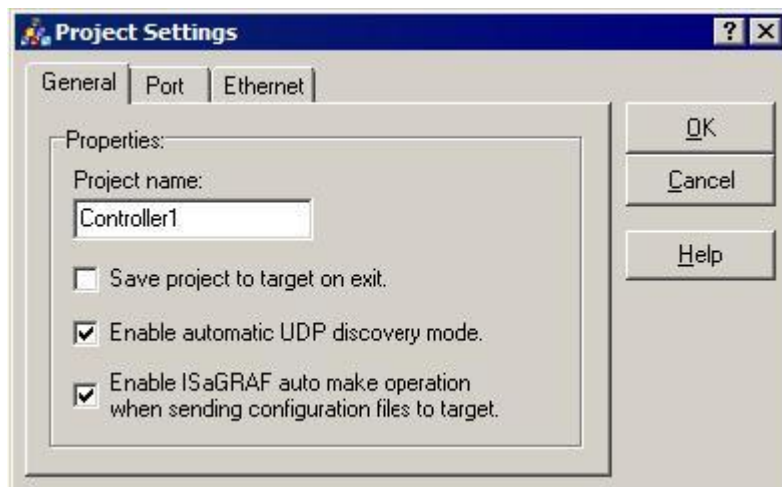
Anytime an upgrade of the ScadaWorks package is done, a Complete Controller Setup... is necessary to synchronize the version of software on the PC and the firmware versions on the controller.

Project Settings Reference

Project options apply to the entire open Project in the Project Manager window. Double click on the project name or click the **Project | Settings** menu.



General Tab



Project Name

This is the name of the ScadaBuilder Project.

Save Project to Target on Exit

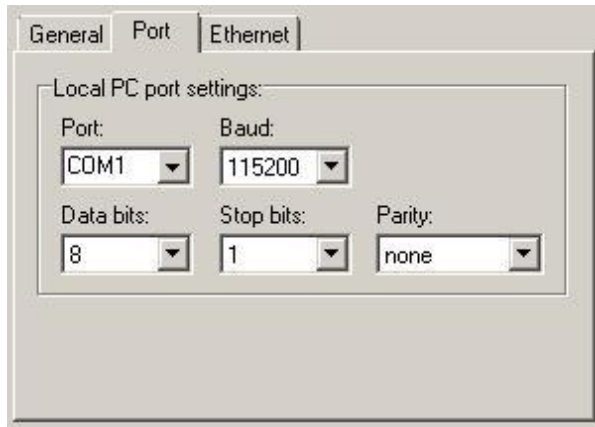
Setting this checkbox will generate a prompt to save (archive) the project to a target node whenever the project is closed. It is checked by default.

Enable ISaGRAF Auto Make

This checkbox, when checked, looks to see if any files in the node have changed. This includes the Node file from ScadaBuilder, any of the ISaGRAF program files or any of the ISaGRAF dictionary (register) files. If they have changed, ScadaBuilder will tell ISaGRAF to do a "Make" before downloading. For downloading duplicate programs

to multiple controllers, if you want the same version to show up in the ISaGRAF debugger on every one, you may wish to disable this option. If you do however, you must do a Make manually when you program changes or the changes are likely not to take affect.

Port Tab

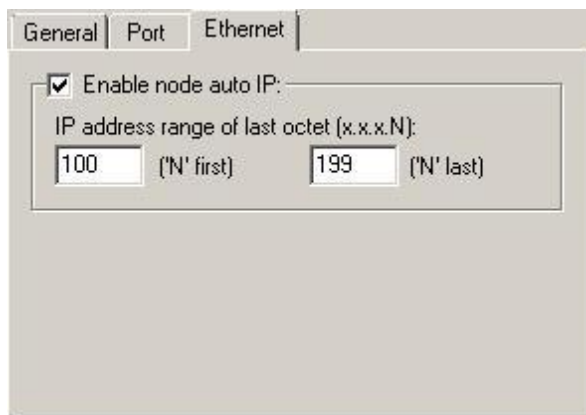


The port settings configure the serial port on your development PC for communications with the target.

The ScadaBuilder Workbench settings must match the settings for the console port (normally COM1) on the target. The typical settings used are 115200 baud, 8 data bits, 1 stop bit and no parity. ICL controllers produced since about February, 2002 have a factory default setting of 115200 baud. Earlier units had a default of 9600 baud. (The baud rate setting can also be changed with a utility called "syscfg" -- contact ICL for details).

For Pinnacle and later controllers, the baud rate is always set to 115200, 8, N, 1.

Ethernet Tab



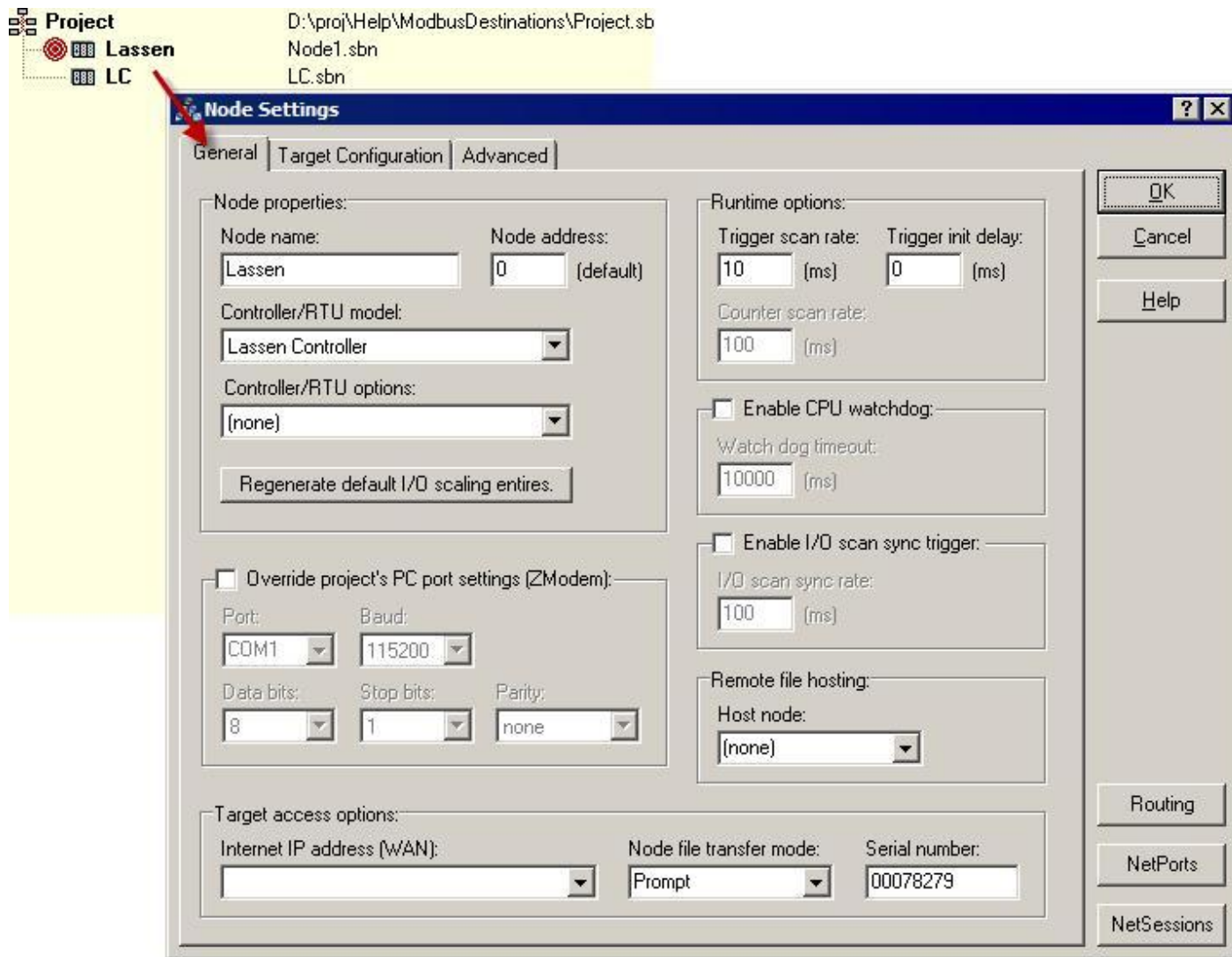
The project Ethernet Tab applies to newly created Pinnacle and later controllers only. When creating a node, an IP address must be assigned. This configurations specifies the IP address that will automatically be applied when the node is created. You can choose to use that IP or change it. Before assigning the IP, ScadaBuilder will ping the network at the new address to make sure no other devices are using that IP. This dialog enables that feature (on by default) and sets the range of addresses that ScadaBuilder will attempt to use.

Node Settings Reference

Node settings apply to the specific Node that you have selected in the Project Manager window. Typically, changes in any of the Node Settings parameters will require at least a Target Configuration download. This can be done by selecting the **Target | Send Startup Config...** menu.

Node Settings - General Tab

This set of parameters identifies the controller and what options are included with it as well as some general configuration parameters.



Node Name

The name of the ScadaBuilder Node. This is also the name by which other Nodes can access configurations within this node. For example:

- IP Address
- Remote File Hosting
- Bricknet register configurations

Node Address

This specifies the default network address for the node. The default address gets used in two ways.

Network Sessions

The default address is used when creating Network Sessions that specify a protocol requiring a network address. This default address is automatically inserted into the session's Network Address dialog box when the Protocol is selected for the session. You can then change this value if desired. This applies to protocols such as Modbus slave, DF1 slave, DNP 3 Slave and Bricknet.

Network Events

The default address is used when creating Network Events on 'another' node that requires a remote reference. This default address is automatically inserted into the event's Address dialog box when the remote node name is selected for the event. You can then change this value if desired. This applies to protocols such as Modbus/TCP Master and Bricknet.



Note: Node Address (default) does not actually affect Network Event and Network Session addresses except at creation time. This parameter can be overridden at any time.

Controller/RTU Model

This selects the controller model that your ScadaBuilder application will run on. A mismatched controller type will still run the program (for testing purposes) however; configurations such as I/O and internal modem or radio options might not work properly.

Some RTU devices may also be configured as a node for register access from protocols like Bricknet.

Controller Options

This selects between any available options for the controller model that your ScadaBuilder application will run on.

Options available are:

- Internal Radios
- Internal Modems
- Internal RS-232/485 interface boards

Regenerate Default I/O Scaling Entries

After a controller type has been changed, the I/O and I/O Scaling Entries are likely to be incompatible with the new controller type. This button allows the new model's scaling types (with the associated I/O modes) to be regenerated and easily mapped to new analog I/O points and registers in the I/O Configuration dialog. The I/O mapping is not restored and must be mapped in the I/O Mapping section by the user.

Trigger Scan Rate

The Trigger Scan Rate setting determines how frequently Triggers are evaluated to see if the trigger condition has become active. Setting the scan rate to a smaller value makes the system more responsive to changes in Trigger conditions, but uses more processing power.

Trigger Init Delay

The trigger initialization delay will inhibit all triggers for the specified time in milliseconds after the application starts. This allows I/O, telemetry and other control decisions to settle so false alarms and events are not generated at startup.

Counter Scan Rate

Only applies to the ICL-4300 controller. Sets the rate at which the pulse counter hardware is read on input boards that support this feature. A faster scan rate allows for higher pulse frequencies, but takes more CPU time.

Enable CPU WDT

This enables the watchdog timer (WDT) system for the controller. The WDT is based on a counter value that is restored by the main loop of the application. A timer interrupt service routine decrements the counter towards 0. If the counter keeps being restored by the main software loop, and never reaches zero, it means that everything is OK. If the counter reaches zero, it means that the main software loop is 'locked up', and the system needs to be reset. The reset will happen automatically.

Watch Dog Timeout

Sets the timeout that is used when the watch dog timer (WDT) is enabled. The main program loop must execute at a faster period than this rate, or else the WDT timer will timeout and reset the controller.

Override Project's PC Port Settings

This option allows the user to associate a serial port with the Node rather than with the Project. In this way, multiple serial ports can be used to download via Zmodem to multiple units without manually switching serial cables. See Project **Port Settings** (see "Port Tab" on page 33) for more information.

Enable I/O Scan Sync Trigger

This feature is used to run the I/O scan faster than the ISaGRAF scan. With this option unchecked, the I/O will be read and written to once every ISaGRAF scan. Enabling this checkbox will break away the I/O scan from the ISaGRAF scan. This feature is used mainly in conjunction with High Speed Logging.

Host Node

Selecting a Host Node allows ScadaBuilder to download application files (the remote controller program) to a master controller which in turn will transfer the files via the File Transfer mechanism in the Network Session. See **Remote Host File Transfers** (on page 281) section for more details.

I/O Scan Sync Rate

Specify the time down to 10 milliseconds for the I/O scan to run independent of the ISaGRAF scan time. This feature is used mainly in conjunction with High Speed Logging.

Internet IP Address (WAN)

This field is used by ScadaBuilder for downloading and accessing the controller from a non-native network (such as the Internet). This configuration is not downloaded to the controller.

This IP address will represent the controller from the WAN side of a firewall, gateway or router while maintaining the controller's "internal" network address configuration.

If the Internet IP Address (WAN) parameter is not defined, then ScadaBuilder will use the Node IP Address as the destination for downloads and file access.

Node File Transfer Mode


You can choose to be prompted for Network or Serial download.


Setting this option to Network will remove the prompt and always attempt a download via the TCP/IP FTP interface.

Setting this option to Serial will attempt to download via serial based on the *Port Settings* (see "Port Tab" on page 33) in the **Project | Setting Port Tab**.

Serial Number

This serial number is necessary for Pinnacle and later controllers to positively identify the controller target on the network. This is especially important when using Loader mode to initialize the controller with a Target | Complete Controller Setup. It allows ScadaBuilder to change the IP address of the controller on the fly to download to it's known address.

To place a unit in Loader mode, press and hold the button on the front panel next to the Status LED and cycle power on the unit. Continue to hold the button until the display show "SB Loader" then release it. To get the serial number from the controller, click the  Discovery button. If the Serial Number is not configured, you will be prompted to do this when doing any kind of download.

From the Target menu, click on Complete Controller Setup... to update all firmware and download the application to the controller. You will be prompted to reboot. After reboot, any program updates can be done from the  Lightning Bolt button which will compile and download any changes to the application.

Target Configuration Tab

Node Settings - Ethernet Tab

The default interface for most controllers is the Ethernet Port. This dialog sets up most of the parameters of how the Ethernet port interfaces with other systems.

In the case of serial ports however, any serial port's TCP/IP parameters both client and server configurations can be accessed from this dialog. The Serial IP section gives an overview of those ports available and those being used by the TCP/IP stack.

Putting a controller on an existing network should be done with care. Most controller situations use a static IP address and this is recommended practice for any controller that is live on a network. If you have a system or network administrator, you should consult them as to what addresses to use.

Configuring a controller with a duplicate IP address on a LAN system where other users are relying on the connection could have dire consequences. It is recommended that you ping the address from a local computer on the same network before connecting the new Controller / Node to make sure that no other device is on the TCP/IP address.

Please see *Appendix A, An Ethernet/Internet Primer for TCP/IP* (on page 649) for more details on configuring your controller and understanding TCP/IP networking in general.

The image shows the 'Node Settings' dialog box with the 'Advanced' tab selected. The 'Ethernet' sub-tab is active, showing configuration for two Ethernet interfaces. For 'Enable Ethernet 1', the 'Specify IP address' option is selected, with fields for IP address (192.168.237.215), network mask (255.255.255.0), and network gateway (192.168.237.254). The 'Include servers in routing table' checkbox is checked. For 'Enable Ethernet 2', the 'Specify IP address' option is also selected, but the fields are empty. The 'Include servers in routing table' checkbox is also checked. On the right side, there are buttons for 'OK', 'Cancel', 'Help', 'Routing', 'NetPorts', and 'NetSessions'. At the bottom, there are buttons for 'Query Target', 'Apply to Target', and 'Restart Target'.

Enable Ethernet

This check box enables the Ethernet port as well as turning on TCP/IP in general for the entire controller. This box must be checked and the required data in this dialog filled out for an Ethernet port to support FTP, HTTP, Modbus TCP, ISaGRAF debugging over Ethernet and Telnet Textual User Interfaces (TUI's). This is always enabled for Pinnacle and later devices.

Obtain/Specify IP address

There are two ways to configure the IP address for the controller when TCP/IP is enabled.

- Obtain IP Address Automatically (DHCP) - If your network has a DHCP (Dynamic Host Control Protocol) server then the IP address can be obtained automatically when the controller starts. This will also configure the network mask and gateway IP parameters. This is useful for applications that act only as network clients, such as sending out emails, transferring files to and from FTP servers or talking to Modbus TCP Slave controllers (with the Modbus TCP Master protocol). In these cases, remote hosts do not need to initiate connections with controller, thus they don't rely on the controller's IP address being "static".
- Specify IP Address - Allows you to manually configure the IP address along with network mask and gateway. This is required when a controller is going to act as a network server (FTP or HTTP) or serving data communications (Modbus/TCP Slave) between nodes. In these cases, the remote hosts must know the IP address of the controller and rely on it being "static" not "dynamic".

Node IP Address

The IP address associated with the Node. IP addresses uniquely identify devices (hosts) on a TCP/IP network. You must pick a valid IP address in order for TCP/IP to be operational.

Contact your network administrator for a proper IP address. (Just picking an IP address at random is not a good idea -- you may disrupt your network!)

If you are the network administrator, then you must set up addressing properly on your network. It is important that each IP address is unique on the network. If your network or device is connected to the Internet, its IP address must be globally unique on the Internet.

If (and only if) your network has no connection to the Internet or any other networks, you may use a set of addresses that are designated for such a purpose. One such set of addresses are of the form:

192.168.x.x

For example, you could use the following IP addresses on a **completely isolated** Local Area Network (LAN) to assign to 5 unique nodes:

192.168.1.1
192.168.1.2
192.168.1.3
192.168.1.4
192.168.1.5

On selected ICL hardware, the last byte of the IP address can be *read from a switch* (see "Use Address Segment From Switch" on page 39). This makes setting the IP address easily accomplished in the field without having ScadaBuilder.



Caution when entering IP address information

If you're not sure of what to fill in here, consult with the system administrator for your network. Do NOT pick random values or use the ones shown!

Use Address Segment From Switch

When this setting is enabled, it will cause the Node to read the last byte of its IP address setting from the onboard address switches.

Notes:

- Only Etherlogic and ScadaFlex Plus controllers support the ability to set the IP address from a switch.
- IP addresses are 4-byte values, with each byte separated by a period character. For instance, 192.168.1.200 is a valid IP address. Each byte of the address has a valid range of 1-254.
- The first 3 bytes of the address are taken from the **Node IP Address** (on page 39) setting.
- If the address switches are set to "0" then the entire Node IP Address setting is used.

Network Mask

The network mask (or "netmask") allows the ICL controller to determine if the remote network node is on the Local Area Network (LAN) or Wide Area Network (WAN).

It is beyond the scope of this document to explain all the details, but we will give a brief example. If the netmask is set to 255.255.255.0, and the controller's IP address is 192.168.1.23, a remote node IP address of 192.168.1.45 is on the LAN and a remote node IP address of 192.168.7.15 is on the WAN. To talk to node on the WAN, the controller must route its message through a gateway. The "zero bits" of the netmask indicate the part of the IP address that can be changed and still have the remote node be on the LAN.

Please see **Appendix A, An Ethernet/Internet Primer for TCP/IP** (on page 649) for more details.

Network Gateway

This is the IP address of the gateway used to access other network nodes beyond the Local Area Network (LAN). This can also be the address of a LAN's Gateway to the Internet including DSL, ISDN and Cable routers.

Network Gateways are often referred to as Firewalls and Routers and MUST be located on the same logical network as the controller.

Domain Name Servers

Specify the primary and secondary domain name servers (DNS) that will be used by controller. Domain name servers are used to resolve host names into numeric IP addresses.

Include Domain Name Servers In Routing Table

Check this box to add the DNS server addresses to the routing table automatically and apply them to this interface. If this box is not checked and there is a second TCP/IP port (i.e. a serial port) and that second interface is set to the default (see *Node Settings - TCP/IP Routing Editor* (see "Routing Button" on page 49)), the system will attempt to do a DNS lookup over the second port instead of using the Ethernet TCP/IP interface which may cause problems.

TCP/IP Max Sockets

This parameter allows more memory and user sockets to be set aside for socket based (TCP/IP) operations such as Email, Modbus TCP/IP/UDP, Telnet TUI and other IP based features. It does not have any affect on FTP or HTTP capabilities.

Set aside a socket for

- for every client you configure in a Modbus TCP Slave Session.
- for a Telnet TUI, configure one for each TUI instance you define on the TUI General Tab
- for each Modbus TCP Master Session with an IP address destination
- for each Email Session.

This parameter may be configured to use up to 64 sockets. Each Socket takes a small amount of memory so there is some trade off with the number of sockets available and the amount of program memory available.

Pinnacle and later controller use a default of 256 sockets to be shared between all TCP/IP interfaces on the box. This does not usually need to be modified.

TCP/IP Max TCP Retry Time

Sets the maximum duration of a tcp retry. If a value greater than 255 seconds is specified, connections never timeout. This is very useful in wireless applications where nodes roam in and out of service.

Include domain name servers in routing table

This option allows the user to automatically add the DNS servers to the Sockets routing table to access the ethernet port for domain lookups. This is especially useful when the default TCP/IP interface is something other than the Ethernet port such as a Dialup PPP or SLIP connection.

Node Settings - ISaGRAF / FTP / HTTP / Admin Tab

The screenshot shows the 'Node Settings' dialog box with the 'ISaGRAF / FTP / HTTP / Admin' tab selected. The dialog has three main tabs: 'General', 'Target Configuration', and 'Advanced'. Under the 'Advanced' tab, there are four sub-sections: 'Enable ISaGRAF:', 'Enable admin login:', 'Enable FTP:', and 'Enable HTTP:'. Each section contains various configuration options like ports, baud rates, timeouts, and checkboxes. On the right side, there are buttons for 'OK', 'Cancel', 'Help', 'Routing', 'NetPorts', and 'NetSessions'. At the bottom, there are buttons for 'Query Target', 'Apply to Target', and 'Restart Target'.

Node Settings

General | Target Configuration | Advanced

Ethernet | ISaGRAF / FTP / HTTP / Admin

☒ **Enable ISaGRAF:**

Controller debug port: Ethernet (dropdown) Controller baud rate: (dropdown)

Local side port (PC side): (dropdown)

☐ **Enable admin login:**

Username: (text box)

Password: (text box)

☒ **Enable FTP:**

Server port number: 21 (text box) Passwords (button)

Client response timeout: 30 (text box) (sec)

TCP/IP settings:

Max sockets: 256 (text box) DNS reply timeout: 2000 (text box) (ms)

Max TCP retry time: 0 (text box) (ms) DNS retry count: 1 (text box)

☒ **Enable HTTP:**

Server port number: 80 (text box) Permissions (button) Web Portal (button)

OK Cancel Help

Routing

NetPorts

NetSessions

Query Target Apply to Target Restart Target

Enable ISaGRAF

If this checkbox is enabled, it means that the selected Node configuration is part of an ISaGRAF project.

ScadaBuilder can be used in conjunction with ISaGRAF. ISaGRAF is used for control logic, and ScadaBuilder is used for communications and other features. When used in this fashion, ScadaBuilder gets its register definitions from the ISaGRAF dictionary.

Controller Debugger Port

This setting determines which port on the target controller is to be used by the ISaGRAF Runtime Kernel for the debugger link to communicate with the ISaGRAF Workbench running on your PC.

If you change this setting, you need to download the changes to the target controller by choosing **Target | Send Complete Controller Setup...**

Pinnacle and later controllers are always set to Ethernet for the debugger port so the options is disabled.

Controller Baud Rate

This setting configures the baud rate of the target controller for the ISaGRAF Runtime Kernel debugger link. The debugger link is used to communicate with the ISaGRAF Workbench running on your PC.

If you change this setting, you need to download the changes to the target controller by choosing **Target | Send Complete Controller Setup...**

Local Side Port (PC side)

This selects the port on the PC side to be used when serially debugging with ISaGRAF. Only available comports on the PC will show up in this drop down box. The baud rate is automatically configured from the debug settings baudrate.

Enable FTP

This enables FTP support on the Node. This feature can only be enabled if there is a TCP/IP interface (such as Ethernet) enabled.

In addition, "Sockets" must be loaded onto the target. This can be accomplished through the ScadaBuilder Workbench by doing a **Target | Send Complete Controller Setup...** when the desired Node is selected in the Project Manager window.



It should be noted that if the controller is to be used as an FTP client (with a Network Session and the FTP protocol defined there) that FTP will be enabled automatically.



When FTP is turned on and made available as an interface, a hidden user name password is generated that follows the ScadaBuilder source files for communicating with the controller. ScadaBuilder uses this account for download the Target configuration, voice files and program files via Ethernet.

FTP Server Port Number

The FTP server port number specifies the TCP/IP port number that is used by the FTP server to monitor (listen) for client connections. The default port number is 21.

Domain Name Server (DNS) Retries

The Domain Name Server (DNS) Retries controls how often the domain name server is queried for an IP address after a failure. The first retry happens after the first try has failed. The system will then wait for double the Retry (ms) parameter before trying again. The time will be double on each successive retry. If the retries are exhausted, a communication failure will be posted according to the local Network Session Configuration. Each try will alternate between the Primary and Secondary DNS servers. In most cases, the status buffer of the client Network Session will show the number of attempts as well.

Client Response Timeout


The FTP client response timeout sets the response timeout for FTP client commands. If the specified time elapses before the expected completion response is received, the system considers the operation to have timed out and failed. The timeout can be disabled by setting the value to '0'.

FTP Passwords Button

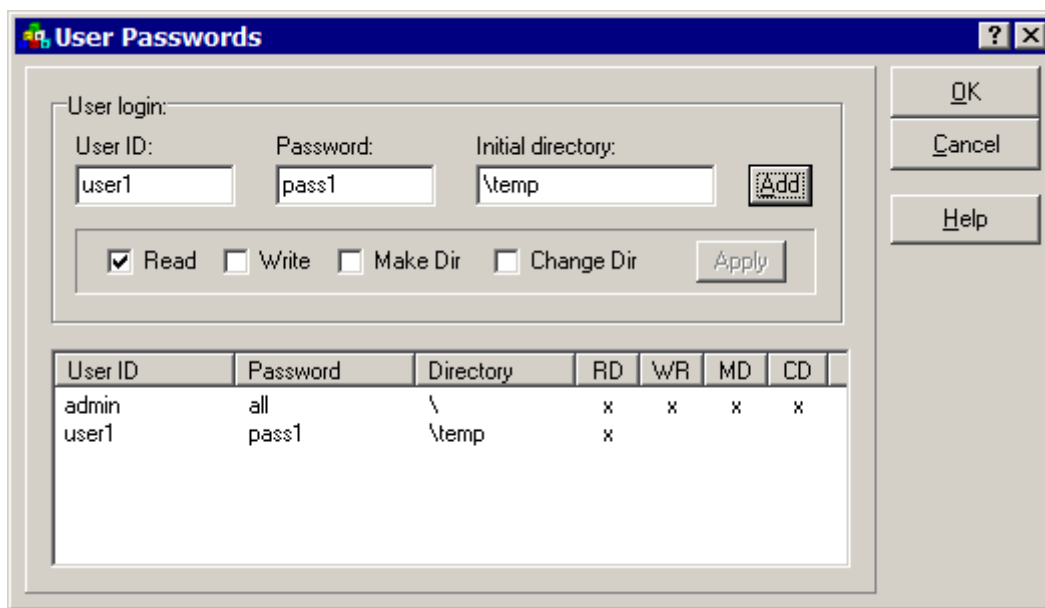
The "Passwords" button opens the User Passwords Editor window. The User Passwords Editor allows you to specify user accounts, passwords and access privileges that are allowed when logging into the FTP server.

FTP User Passwords Editor

The User Passwords Editor allows you to specify user accounts, passwords and access privileges that are allowed when logging into the FTP server.

To start, enter a user ID (this is the account) and password for the user in proper fields. Next, specify the initial directory you wish the user to start in when they log in. Also, set the check boxes to specify the privileges that will be allowed for the user when they login. Click the "Add" button to add the entry to the list. Repeat for as many entries as you wish. To find out more about a particular option, click the  button and then click the option.

Let's look at a specific example:



The first entry "admin" will cause the user to initially log into the root directory and have unrestricted privileges. The second entry "user1" will log into the '\\temp' directory with read only access.

Once an entry has been put on the list, you can change its order or delete it by right clicking on the item. The "Apply" can be used to change the privileges of a selected entry.

NOTE: If no entries are added to the list a default "anonymous" account will be created that allows logging into the root directory with read only access.

User ID

Specifies the user ID to use for the entry.

Password

Specifies the password to use for the entry. Use a single '*' character if no password is required for the entry.

Initial Directory

Specifies the initial directory that is used when the user logs in. Use a single '\' character to represent the root directory.

Read Checkbox

This checkbox indicates that the user can read files from the file system.

Write Checkbox

This checkbox indicates that the user can write and delete files from the file system.

Make Dir Checkbox

This checkbox indicates that the user can create directories on the file system.

Change Dir Checkbox

This checkbox indicates that the user can change directories on the file system. Specifically to a "parent" directory of the initial directory. The user can always move to a subdirectory under their specified initial directory.

Apply Button

The "Apply" button applies the currently selected privileges to and selected (highlighted) entry in the list.

Enable HTTP

This enables HTTP support on the Node. This feature can only be enabled if the "Enable TCP/IP" checkbox on the "Ethernet / Serial IP" Tab has been selected. In addition, "Sockets" must be loaded onto the target. This can be accomplished through the ScadaBuilder Workbench by choosing **Target | Send Sockets** on the main menu when the desired Node is selected in the Project Manager window.

HTTP Server Port Number


The HTTP server port number specifies the TCP/IP port number that is used by the HTTP server to monitor (listen) for client connections. The default port number is 80.

HTTP Permissions Button

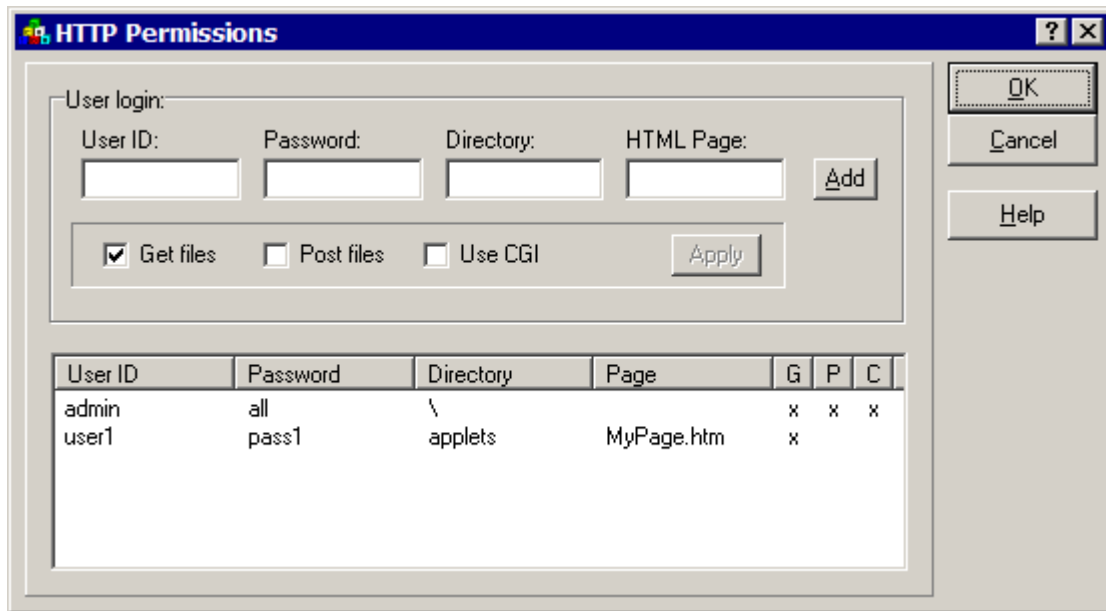
The "Permissions" button opens the HTTP Permissions Editor window. The HTTP Permissions Editor allows you to specify user accounts, passwords and access privileges that are allowed when logging into the HTTP server.

HTTP Permissions Editor

The HTTP Permissions Editor allows you to specify user accounts, password and access privileges that are allowed when logging into the HTTP server. Start directories and HTML page links may also be specified.

To start, enter a user ID (this is the account) and password for the user in the proper fields. Next, specify the initial directory you wish the user to start in when they log in. If you specify an HTML page (optional) then an HTTP link to that page will be added to the **Tools | URL list** menu. Set the check boxes to specify the privileges that will be allowed for the user when they login. Click the "Add" button to add the entry to the list. Repeat for as many entries as you wish. To find out more about a particular option, click the  button and then click the option.

Let's look at a specific example:



The first entry "admin" will cause the user to initially log into the root directory and have unrestricted access. The second entry "user1" will log into the 'applets' directory (off the root) with only get file privileges. The HTTP link "<ip_address>/applets/MyPage.htm" will be added to the URL list.

Once an entry has been put on the list, you can change its order or delete it by right clicking on the item. The "Apply" can be used to change the access privileges of a selected entry.

NOTE: If no entries are added to the list then a default account will be created that will allow anyone to log into the root directory with unrestricted access.

User ID

Specifies the user ID to use for the entry. Use a single '*' character for when the client logs in without a user name.

Password

Specifies the password to use for the entry. Use a single '*' character if no password is required for the entry.

Directory

Specifies the initial directory that is used when the client logs in. Use a single '\' character to represent the root directory.

HTML Page

If an HTML page (optional) is specified then an HTTP link to that page will be added to the **Tools | URL List...** on the main menu. Each link on the URL list will be displayed in the format:

"<ip_address>/<directory>/<page.htm>"

You can then click on the HTTP link to launch your web browser and load the specified page. This is typically used with the ErgoView option to provide a convenient way to access web pages that have developed and loaded on the system.

Add Button

The "Add" button causes the entry to be added to the user list.

Get Files Checkbox

This checkbox indicates that the user may get files from the file system.

Post Files Checkbox

This checkbox indicates that the user may post files to the file system.

Use CGI Checkbox

This checkbox indicates that the user may use CGI.

Apply Button

The "Apply" button applies the currently selected privileges to the selected entry in the list.

Enable Admin Login

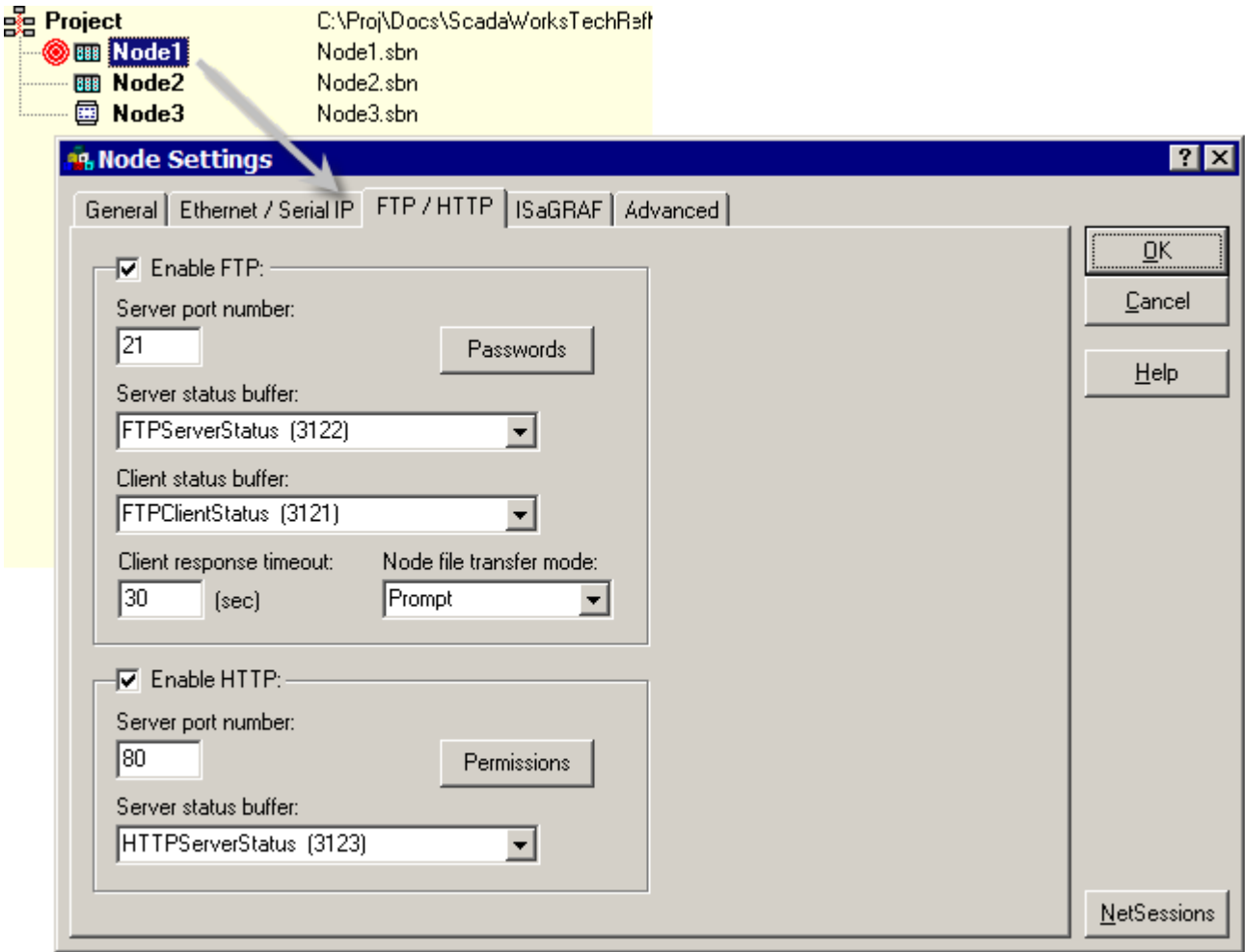
The administration login is used to secure the controllers administration and maintenance ports for things such as downloading and configuring the controller after the application has been release. This can be used for applications that require a total secure lock down of the controller.

Web Portal Button

This section is used to setup the initial "static" accounts used by the ICL User Portal which can be accessed from any browser by the controller's IP address. For more details on setting this up, see the *User Portal (Web Interface)* (on page 423) section.

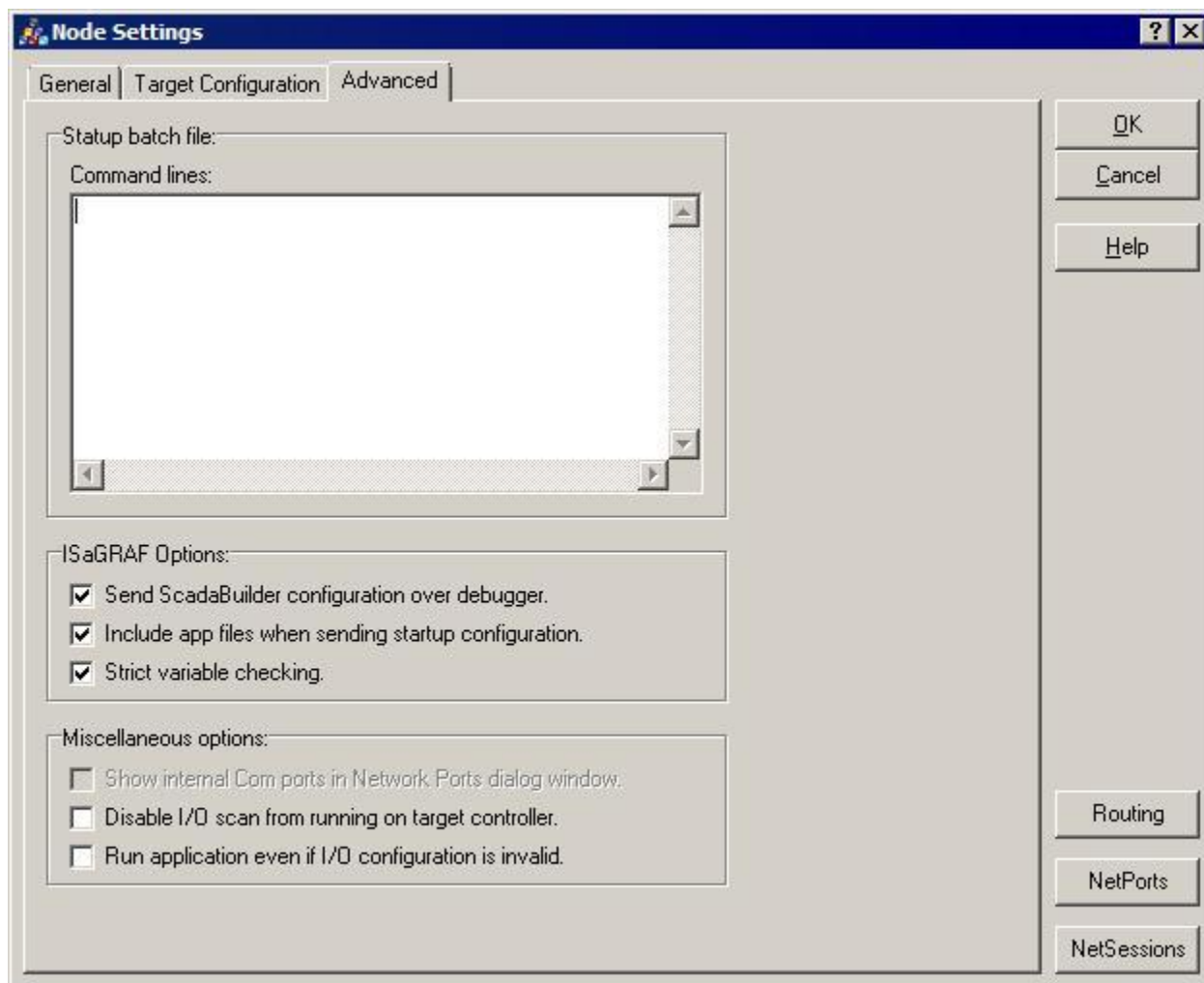
Node Settings - FTP/HTTP Tab

All FTP and HTTP server configuration are done in this tab.



Node Settings - Advanced Tab

Most options in here are for advanced users only. Contact technical support should you find yourself needing the features here.



Startup Batch File Command Lines

For advanced users only.

These are DOS batch file commands that are to be included in the batch file that starts the ScadaBuilder Node. This allows you to do any special processing before the ScadaBuilder Virtual Machine or ISaGRAF Kernel and the Node configuration files are loaded.

This option is not available for Pinnacle and later controllers.

Show Internal Com Ports in the Network Ports Dialog Window

Advanced users only.

If enabled this will show and enable use of internal communication ports on those controllers (such as EtherLogic and ScadaFlex Plus) that use internal serial I/O processors. This allows access to some advanced features via Modbus. See the appropriate hardware reference guide for details and a Modbus register map.

Disable I/O Scan From Running On Target Controller

If this setting is checked, I/O updating will be turned off. This can be useful as a troubleshooting aid to isolate the controller from what the real I/O is doing.



For normal operation, do not disable the I/O scan

Include ISaGRAF Application Files

When checked (default), the ISaGRAF application files will also be sent when the startup configuration (see **Target | Send Complete Controller Setup...** menu selection) is sent to the target node. By first performing a make (see **Tools" | ISaGRAF Make...** menu selection) to compile any application changes, this allows the user send all of the updated program files that are required to run the application (without having to launch the ISaGRAF development/debugger tools).

Enable ISaGRAF Strict Variable Checking

When enabled (default), strict checking will not only insure that the endpoints of a referenced block are valid ISaGRAF variables, but that all the variables in the block are valid as well.

For example, with strict checking enabled, the ScadaBuilder Workbench will not let you attach a block with a missing variable to a "delta" Trigger.

Send ScadaBuilder Configuration Over Debugger

This option is left checked most of the time. It can be used to recover some memory from the system at the cost of a little convenience.

Checked

The ScadaBuilder node file "node.sbr" which is stored in the <node name> directory will be compiled into the ISaGRAF program image during a Make which allows it to be transferred to the controller when an ISaGRAF Debugger download is done. This convenience costs application memory by placing the node.sbr file as a binary resource within the ISaGRAF program. The larger the file, the less memory available to the ISaGRAF application.

Unchecked

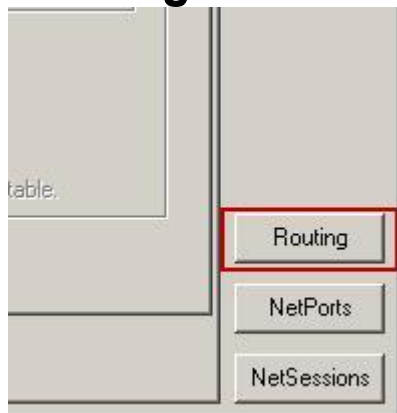
The ScadaBuilder file "node.sbr" will not be compiled into the ISaGRAF program when a Make is done. The ScadaBuilder configuration must then be downloaded using the **Target | Send Startup Config...** menu or the Target



button. Changes done in ScadaBuilder will not be updated until the Target download operation is performed.

The advantage however, is a significant gain in program memory. This is particularly useful in combination with other memory intensive features such as HTTP and FTP when resources are at a premium.


Routing Button



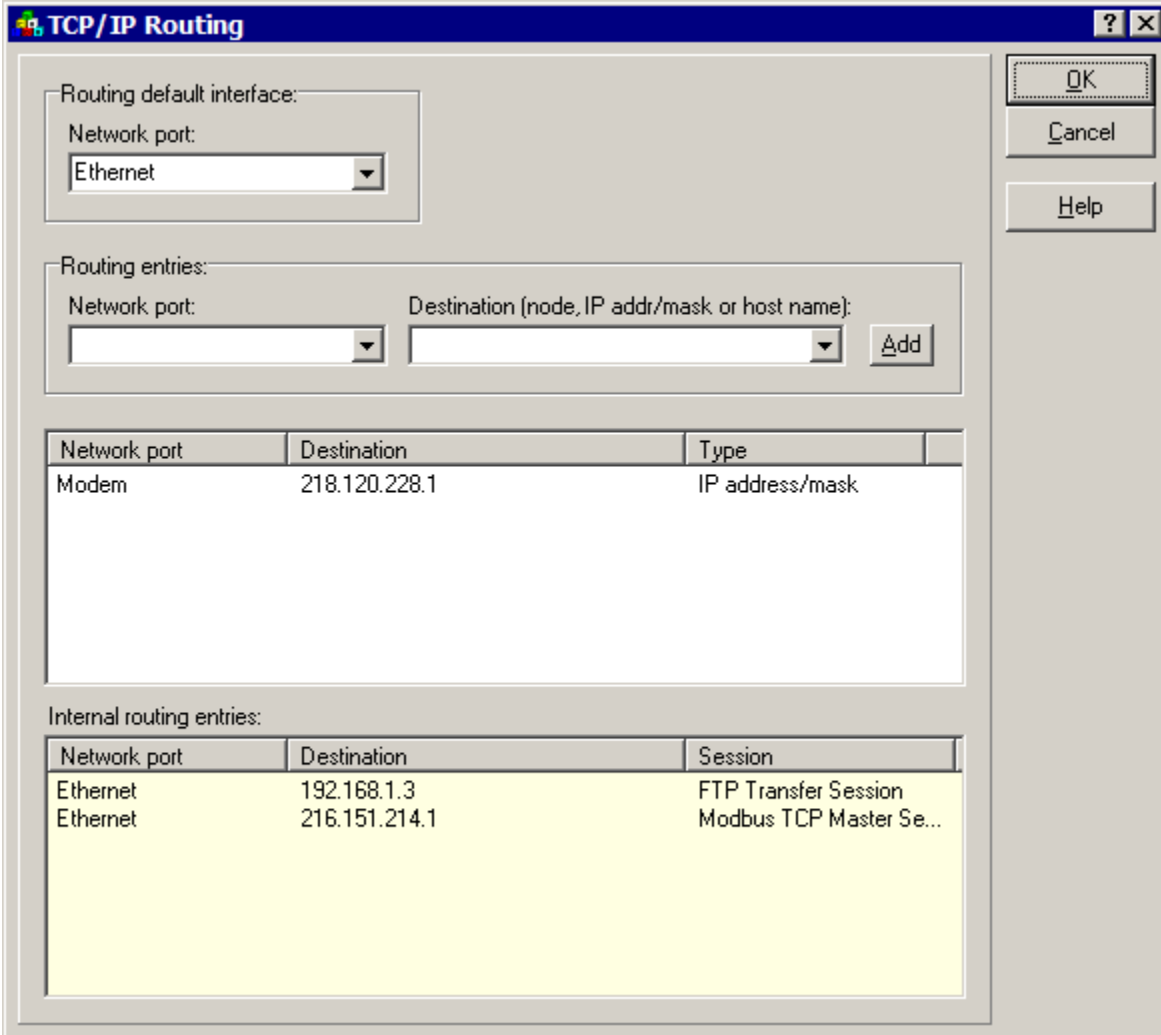
The Routing button takes you to the TCP/IP Routing editor.

The TCP/IP Routing Editor allows you to create entries that determine which ports should be used when sending data in order to reach the desired host. This only needs to be done if you have configured your controller to use multiple TCP/IP interfaces (such as both an Ethernet and PPP serial port).

To start, select the default interface. This is the interface port that will be used to send all transmission to IP addresses that are not otherwise specified in the routing table.

Next, enter the network port that will be used to communicate with the destination. Then, select the destination type (this can be either a host name, numeric IP address or other node that is part of the project). Finally, enter or select the destination that is appropriate for destination type. Click the "Add" button to add the entry to the list. Repeat for as many entries as you wish. To find out more about a particular option, click the  button and then click the option.

Let's look at a specific example:



The dialog box is titled "TCP/IP Routing". It contains the following sections:

- Routing default interface:** A group box containing a "Network port:" label and a dropdown menu currently set to "Ethernet".
- Routing entries:** A group box containing a "Network port:" dropdown menu, a "Destination (node, IP addr/mask or host name):" text field, and an "Add" button.
- Routing entries table:** A table with three columns: "Network port", "Destination", and "Type". It contains one entry: "Modem" with destination "218.120.228.1" and type "IP address/mask".
- Internal routing entries:** A group box containing a table with three columns: "Network port", "Destination", and "Session". It contains two entries highlighted in yellow: "Ethernet" with destination "192.168.1.3" and session "FTP Transfer Session", and "Ethernet" with destination "216.151.214.1" and session "Modbus TCP Master Se...".

Buttons on the right side include "OK", "Cancel", and "Help".

The default routing entry is the "Ethernet" port. Any transmission to an IP address that is not specified in the routing table will go out the Ethernet port.

An entry was made specifying that the Com3 network port interface should be used when communicating to the "Modbus Slave Node". An entry like this typically means that Modbus/TCP Master has been configured to run on Com3 and it is connected to a remote node that is running Modbus/TCP Slave. The IP address that was specified for the Modbus Slave Node (remote node) will become the destination IP address.

Once an entry has been put on the list, you can change its order or delete it by right clicking on the item.

The addresses in the yellow area are configured automatically when a Network Session uses a TCP/IP Network Port of any kind for client (master) operations. These entries are done by the system and are not editable.



A more complete tutorial on Ethernet with a detailed description of routing is contained in Appendix A (see "Appendix A, An Ethernet/Internet Primer for TCP/IP" on page 649) at the back of this manual. The tutorial is entitled "The Ethernet/Internet Primer for ICL Controllers."

Routing Editor - Default Network Port

Specifies the network interface port that will be used to send all transmission to IP addresses that are not otherwise specified in the routing table.

Routing Editor - Entry Network Port

Specifies the network port that will be used to communicate with the destination. Only network ports that have been configured (enabled) to use TCP/IP will show up in the list.

Routing Editor - Entry Destination Type

Specifies the destination type for the remote host/node. This allows the user different methods for entering the destination. The destination can be entered as either as a project node name, numeric IP address/mask or host name.

Routing Editor - Entry Destination

Specifies the destination for the routing entry. This control will morph depending on the selection of the destination type. There are three different destination types as listed below:

- Node Name - This references a node or controller that is part of the project. The IP address that was configured for the selected node will become the destination IP address. The selected remote node cannot use DHCP as it must have a static IP address.
- IP Address/Mask - This is used to specify a numeric IP address or range of addresses for the destination.
- Host Name - This specifies the name of remote host that is the destination. DNS entries must be specified for the local node so the host name can be resolved.

Routing Editor - Internal Routing Entries

Entries are placed in this list as outgoing sessions that use TCP/IP ports are configured. Internal Routing Entries are not editable.

Network Port	Specifies the physical Network Port the route (destination I/P address) will use.
Destination	I/P Address the Controller / Node is trying to contact over the above Network Port Interface.
Session	Network Session that will use the route (destination I/P).

NetPorts Button

Brings up the Network Ports dialog. This is the same as the **Setup | Network Ports...** menu.

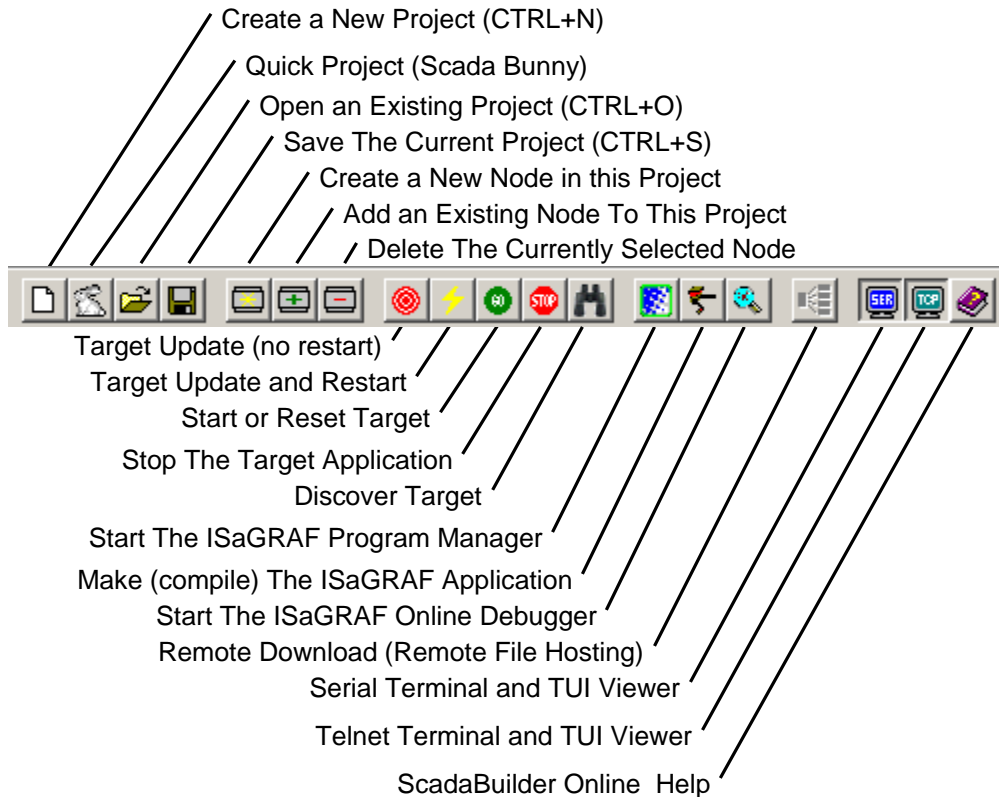
NetSessions Button

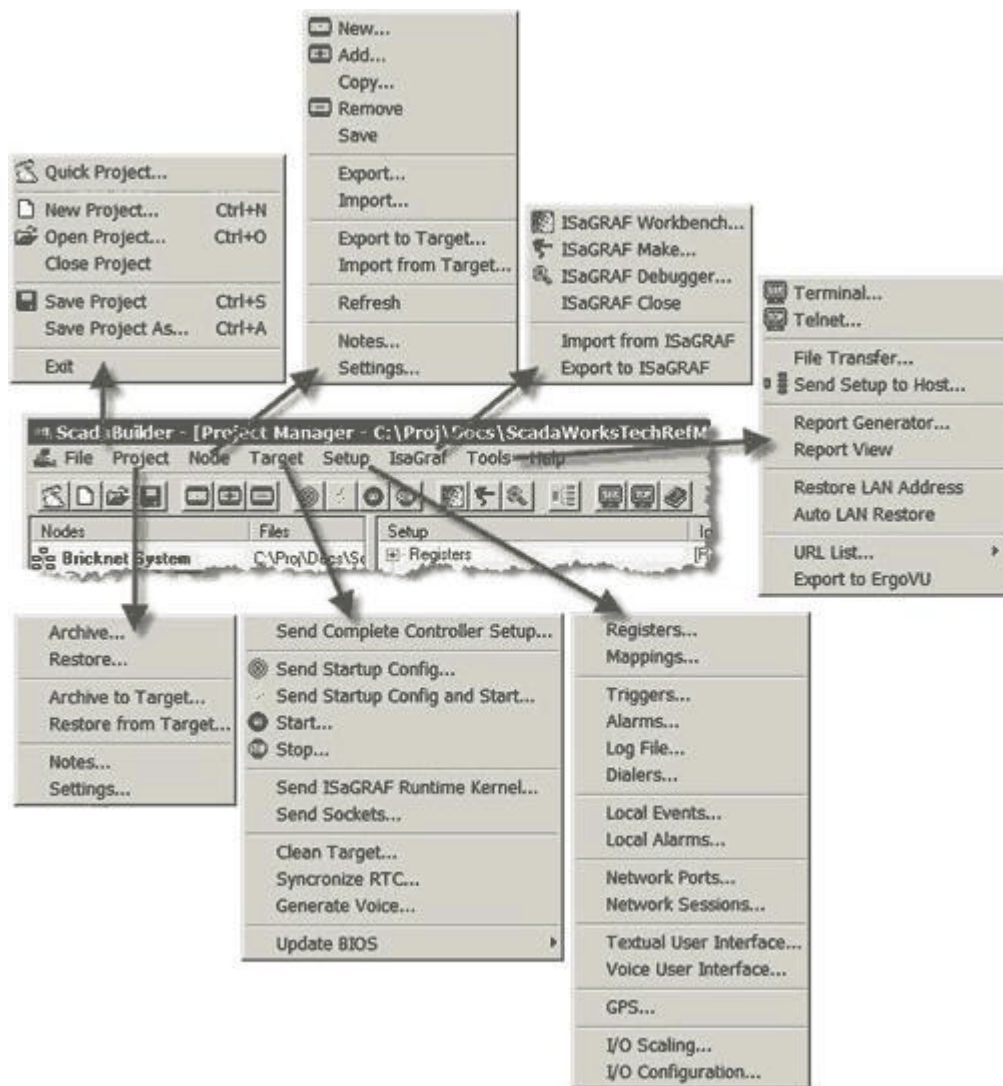
This button brings up the Network Sessions dialog. This is the same as clicking on the **Setup | Network Sessions...** menu.

SECTION III

The ScadaBuilder User Interface

The ScadaBuilder Program Manager gives an abundance of tools to provide you with the most connectivity to your controller(s) and the most visibility into your process. There are multiple ways to get data in and out of the box. Many of the tools for doing so are right in the toolbars and menus.





In This Section

File Menu	54
PROJECT Menu	55
NODE Menu.....	56
TARGET Menu	57
SETUP Menu.....	62
ISaGRAF Menu	63
TOOLS Menu	64

File Menu



Quick Project

The Quick Project icon (also known as the ScadaBunny) is to start a quick project. This will build the project and node while prompting you through the steps to get it done. It is the easiest way to create a new controller application.



New Project

Starts a new project in a new directory. It does not create a usable application, only a template for holding more nodes.



Open Project

Opens an existing project. It allows the user to browse to their existing projects and open them.

Close Project

Closes the current project and prompts if there are any changes to be saved in the nodes or the project.

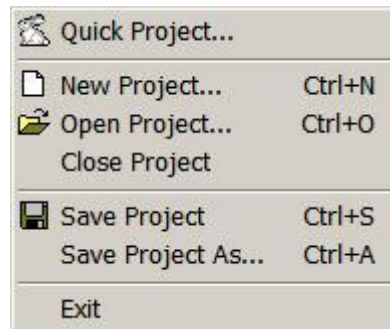


Save Project

Save the current project to the .SBP file. This does not save the individual nodes in the project, only the project configuration.

Save Project As...

This allows you to save a project some where else on your computer as a working copy. If you are looking for backups, use the **Project | Archive** menu and save off the resulting ZBP file.



PROJECT Menu

Archiving and Restoring Source Code To and From The Controller

Archive...

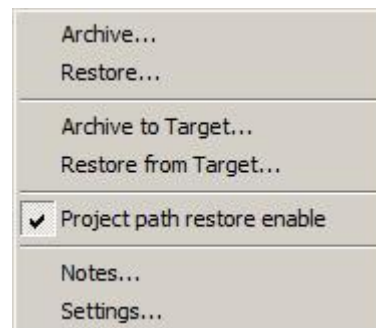
Allows you to archive a project to backup and restore at a later time. Archiving a project zips everything in the project directory into one file (except other project archives). Archives may be password protected. If you do not want a password then simply hit enter when prompted.

Restore...

Allows you to create a new directory for a project and restore a previously "Archived" project to that path.

Archive to Target...

Archives the current project and then downloads it to a controller's file system either serially (Zmodem) or if available, over FTP. This allows others to restore from the controller so they know they have the latest source code.





Archiving to the target controller is only way to be able to upload the code from the controller later on. If this step is not done, then someone without the source code CANNOT modify the program. They can only start anew.

Restore From Target...

Allows project archives to be retrieved from the controller if they have been placed there. If they have not, there is no way (unless they downloaded a node archive to the unit) to retrieve to source code. It is not stored by default.

Project Path Restore Enable

If checked on a project restore, this option will tell ScadaBuilder to try and restore to the directory where the project archive being opened resides. Otherwise, it will use the last known location.

Notes...

Allows notes to be place with this project. When a report is generated (see the **Tools menu** (on page 64)), these notes will show up close to the head end of the project documentation.

Settings...

These are basic settings that govern how the project downloads and compiles programs for you automatically. See **Project Settings** (see "Project Settings Reference" on page 31).

NODE Menu



New...

Add a new controller or RTU Node to this project. You will be prompted for the controller/RTU type as well as any communication option it may have.



Add...

Add an existing Node to the project. It must already reside in the project directory.

Copy...

Copy the current node and all of its registers and source code into a new node in this project.

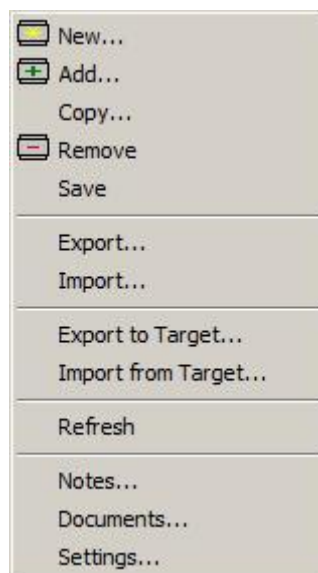


Remove...

There are two styles of removal, 1) removes the node from the project tree but leave the files in place (so that you may Add it later on). The other 2) will prompt you to delete all files meaning the node and its directory will be gone with no undo.

Save

Saves the current .SBN file (node) that has the target icon next to it. If you exit from ScadaBuilder, you will be prompted to save as well.



Importing and Exporting Source Code To and From The Controller

Export...

This allows you to Export a node and zip up everything about that Node and transfer it to another project. This produces a .ZBN file to import later on. It can be password protected. If you do not want to use a password then simply hit enter when prompted.

The only caveat to this is if there are Bricknet links to other nodes in the project. They will have to be manually reconciled or deleted in a new destination project.

Import...

Imports a .ZBN file into the current project that was previously Exported.

Export to Target...

Exports then downloads to the controller the resulting .ZBN for later recovery. This only downloads the information about the selected node (the one with the target icon next to it). The .ZBN file will be downloaded serially (Zmodem) or via FTP if available.

Import from Target...

Imports a previously Exported to Target .ZBN file. If the previous user did not place the file on the controller (and there is no project archive on that controller), then there is no way to modify the program if you do not already have it.

Notes

Notes are good place to do things like revision history and documentation. These notes will show up (if selected) at the head of the Node documentation from the Report Generator. See **TOOLS Menu** (on page 64) Report Generator for more details.

Documents...

The User Portal has the ability to display documents loaded on the controller if the user has permissions to do so. From this option, documents from the local hard disk can be downloaded when a Complete Controller Setup... is done. These documents are loaded onto the controller in the c:\Webroot\Docs directory. Any document in this directory may be loaded from the User Portal web interface.

Node Settings...

For more detail on the Node Settings Dialog see

Node Settings - General Tab (on page 34)

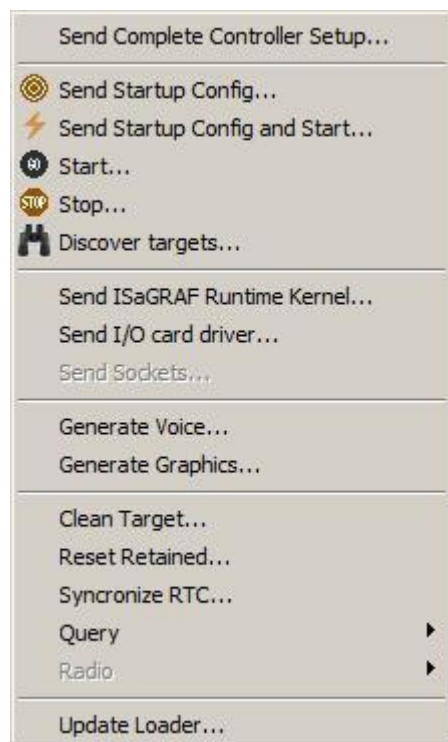
Node Settings - Ethernet / Serial IP Tab (see "Node Settings - Ethernet Tab" on page 37)

Node Settings - FTP/HTTP Tab (on page 46)

Node Settings - Advanced Tab (on page 47)

TARGET Menu

All items in the Target Menu are used to update a controller in some way. When an application is modified and compiled, at least one option on the Target menu will be used to get the configuration, program or both down to the controller.



Send Complete Controller Setup...

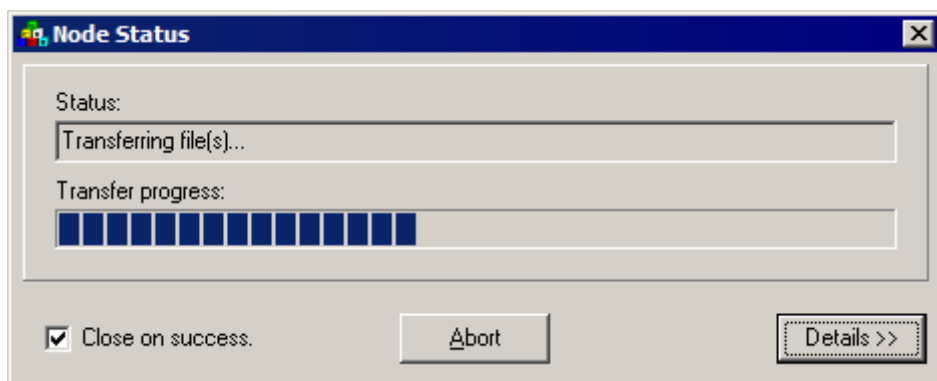
This option is used whenever the version of the ScadaWorks workbench and the runtime kernel on the controller are out of sync or if a new application is being downloaded to a controller. This option causes a download of all files required to run on the controller.

For older controllers...

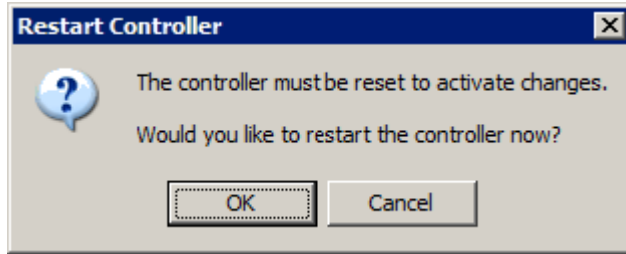
Target | Send Complete Controller Setup... is always done over a serial link with a null modem cable from the PC to Com1 of the controller.

For Pinnacle and later controllers, the controller must either be in loader mode, or it must already have a copy of the currently open node already downloaded to the controller to allow Complete Controller Setup... to work. If neither of these conditions are met then the download will be blocked for security reasons.

- 1 If a voice feature (*Dialers* (see "Using Alarm Dialers" on page 169) or *VUI* (see "Using the Voice User Interface" on page 183)) is used in your application, you will be prompted to regenerate the voice files for download. If not, the download dialog will appear.
- 2 If a voice feature is used, the VOICE.SBW will be downloaded to the unit.
- 3 The download will attempt to do an Application Make (Compile) automatically.
- 4 If the Make was successful, the system will attempt to download the files of the application.




- 5
- 6 At the end of the process, you will be prompted to restart the controller, Click OK.



7

8 The application should start up without any errors.



All of the Target menu items unless otherwise noted may be utilized over FTP for remote downloads and updating. **Node Settings - FTP/HTTP Tab** (on page 46).

Exceptions are **Target | Update Bios...**, **Target | Update Loader...** and **Target | Send Complete Controller Setup...**



Send Startup Config...

Downloads just the application files for the controller. This may be done over FTP or over a serial port. Changes in the Node Settings dialog in ScadaBuilder are not updated from this menu item or the Target button. This option does not restart the application but will stop the application if downloading over a serial link.



Send Startup Config and Start...

This option does the same thing as the **Target | Send Startup Config...** option above but restarts the application automatically after the downloads complete.



Start...

Restarts the application whether it is currently running an application or it is in the Stopped state.



Stop...

This option shuts down the application/

Send ISaGRAF Runtime Kernel...

Updates the ISaGRAF runtime Kernel and synchronizes the versions with ScadaWorks installed on the computer. This option is also done from the **Target | Complete Controller Setup...** option above.

Send Sockets...

Updates the TCP/IP drivers on the controller. It is recommended that this be done over a serial connection though FTP updates are possible. This option is also done from the **Target | Complete Controller Setup...** option above. This is not used for Pinnacle and later controller and will be disabled.

Clean Target...

Removes all application related data from the controller preparing it for a new application. This option is also done from the **Target | Complete Controller Setup...** option above.

Reset Retained...

Pinnacle and later controllers. Resets the retained registers on a unit. If the registers have an initial value, then this value will be reapplied to the registers. If not, the register will be set to false, 0, 0.0 or an empty string "" for the appropriate data type. The application will be restarted to make this take effect.

Generate Voice...

Generates the voice file, does an application Make if necessary and downloads the appropriate voice files to the controller. The controller must have an ICL dialup option to utilize this feature. This option is also done from the **Target | Complete Controller Setup...** option above.

Query...> (Pinnacle and later controllers only).

Query Format

Returns the format version of the node configuration and also the version of the ISaGRAF kernel.

Query Memory

Returns the amount of memory available to the application and total memory on the controller.

Query Disk Space

Returns the total amount of available disk space on the controller.

Update Loader...

Pinnacle Series only. Unit must be in Loader Mode to update the loader. There are actually two loaders, both will be updated from this option to the latest version installed with the ScadaWorks package.

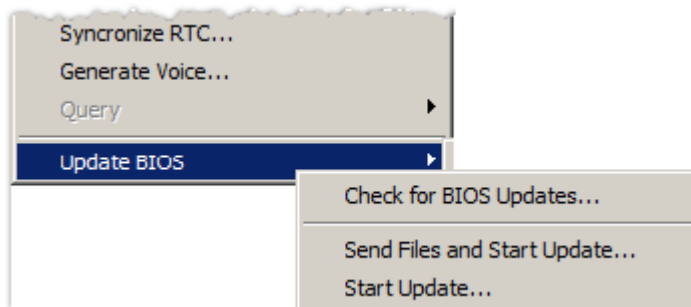
Update Bios...

Non-Pinnacle controllers. As controllers progress, they sometimes need the level of software below the ISaGRAF runtime kernel to be updated. This must be done over a serial link and requires user interaction.



Contact ICL Technical Support (see "Copyright Notice" on page 2) before updating the BIOS to see if this is a necessary step for you.

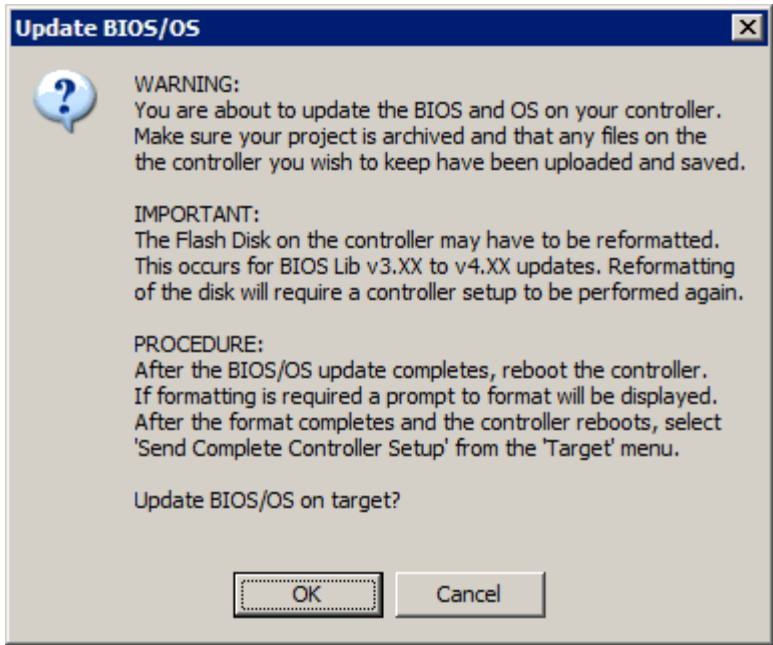
Do not interrupt power to the controller during this process.



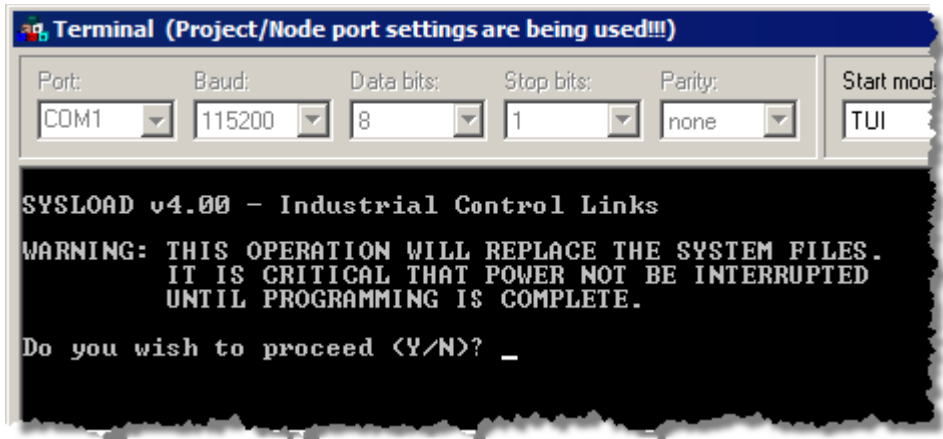
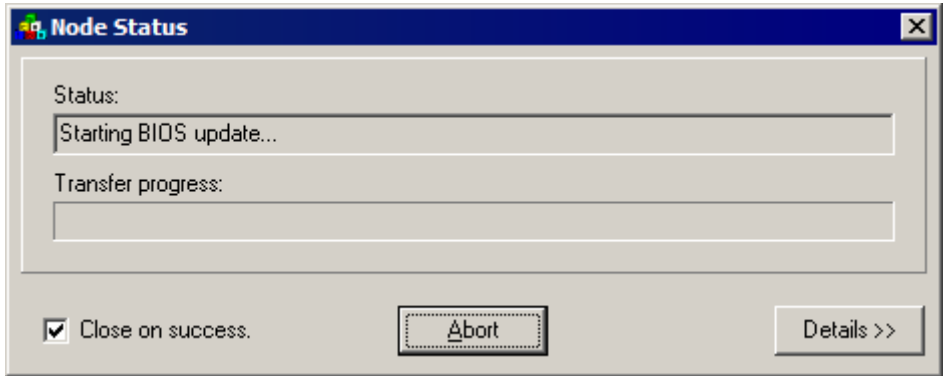
Check For Bios Updates...

This option requires an Internet connection and will go to the ICL web site and download the latest BIOS files available for all controllers.

Send Files and Start Update...



This will send the utility and BIOS files down to the controller and begin the BIOS update. The Terminal window will open and you will be prompted through a couple of questions.



Click OK and hit "Y" to start the process.

Wait for the process to finish.

```
SYSLOAD v4.00 - Industrial Control Links

WARNING: THIS OPERATION WILL REPLACE THE SYSTEM FILES.
         IT IS CRITICAL THAT POWER NOT BE INTERRUPTED
         UNTIL PROGRAMMING IS COMPLETE.

Do you wish to proceed (Y/N)? Y

Scanning system file... Success.
  File Lines: 6263

Loading system file...
  Percent Complete *****..... 40%_
```

```
Erasing flash memory (OS and Disk)... Success.

Writing RAM Image to flash memory...
  Percent Complete ***** 100%

Restoring configuration parameters to flash... Success.

Verifying RAM Image in flash... Success.

System load complete (15.0 seconds).

*****
* RESTART THE SYSTEM (cycle power or press reset button) *
*****

C:\>
```

Reset the controller by the Reset button on the controller or by cycling power.

You will see the system copy a few files. The controller should now be ready to accept new firmware.

Close the terminal and click on the **Target | Send Complete Controller Setup...** menu to complete the upgrade process.

Start Update...

If the files have already been sent to the controller, you may start the update at any time. This is in case the updates failed the first time through.

SETUP Menu

The setup menu simply is a navigation tool to get to the right record to setup. Follow the links for more information.

Registers...
Mappings...
Triggers...
Alarms...
Log File...
Trends...
Dialers...
Local Events...
Local Alarms...
Network Ports...
Network Sessions...
Textual User Interface...
Voice User Interface...
Text Message Interface...
Programs...
Gas Flow...
GPS...
I/O Scaling...
I/O Configuration...

Registers (on page 103)

Mappings (see "Mappings Reference" on page 493)

Using Triggers

Using Alarms (on page 159)

Using Log Files -- Data and Alarm Logging (on page 77)

Setting Up a New Trend (on page 456)

Using Alarm Dialers (on page 169)

Using Local Events (see "Using Local Alarms" on page 165)

Using Local Alarms (on page 165)

Using Network Ports (on page 198)

Using Network Sessions (on page 213)

Textual User Interface (TUI) (on page 385)

Using the Voice User Interface (on page 183)

Text Message Interface (TMI) (on page 483)

C DLL Programming Environment

AGA Gas Flows

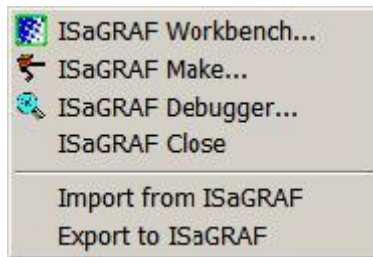
Global Positioning Satellite (GPS) Interface (on page 497)

I/O Scaling

Using I/O Channels and Mapping Registers (on page 129)

ISaGRAF Menu

The ISaGRAF Menu provides access to the ISaGRAF program development tools and utilities that are useful for controller program testing and maintenance. Some of the ISaGRAF menu elements have equivalent “hot keys”. For these elements, the hot key icons are pictured before the element name.



ISaGRAF Workbench

Select this menu item to bring up the ISaGRAF program development workbench. Use this tool to create and edit control programs in the ISaGRAF development environment.

ISaGRAF Make

Select this menu item to compile (“make”) a ScadaWorks program into an executable program that can be downloaded to the controller.

ISaGRAF Debugger

Select this menu item to bring up the ISaGRAF program debugger to download a ScadaWorks program and monitor its execution with ISaGRAF real-time debugging tools.



The menu items below are supported only for legacy purposes. The ISaGRAF dictionary gets read when ScadaBuilder comes into focus and written when ScadaBuilder goes out of focus or a configuration necessary to ISaGRAF changes.

Import ISaGRAF

Select this menu item to load the ISaGRAF dictionary from the currently selected target node.

Export ISaGRAF

Select this menu item to save the ISaGRAF dictionary to the currently selected target node

TOOLS Menu

Terminal...

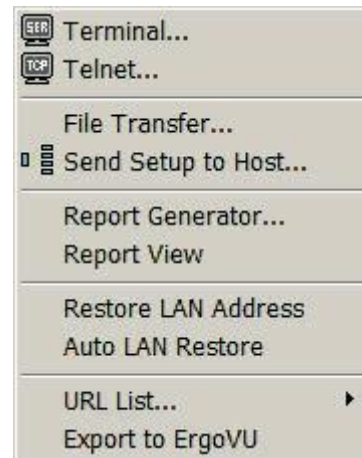
Opens the serial port terminal in ScadaBuilder. It is used for Console interaction and TUI viewing and interfacing. The default setting for the Terminal and ICL controller consoles are 115200 baud, 8 data bits, no parity and 1 stop bit.



If you drag and drop a file from Windows Explorer, the file will be downloaded over the node's preferred interface (FTP or Zmodem).

Telnet...

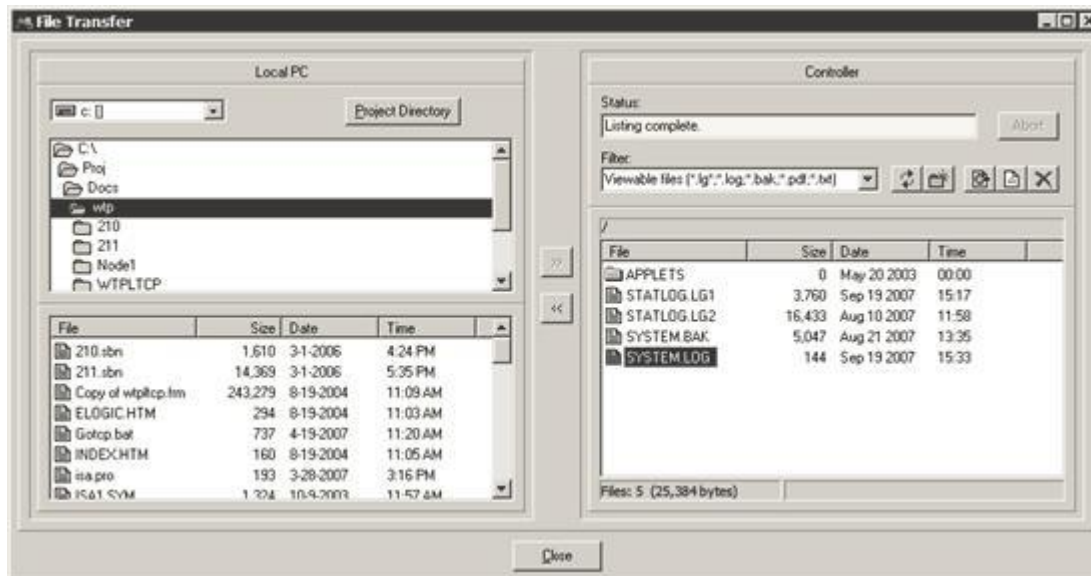
Opens a TCP/IP port typically to the Node's IP address on port 23 (that is configurable in the TUI interface). The telnet terminal is ideal for TUI viewing remotely over Ethernet or dialup.



File Transfer...

This opens a File Transfer window that can work either over serial (with the controller stopped) or with the FTP interface if configured. See **Node Settings - FTP/HTTP Tab** (on page 46) for more information on configuring FTP.

The File Transfer Window




This window allows you to do file maintenance over the serial port or over FTP (Ethernet). You can view upload, or download any file you wish from the controller. To use over FTP, you must have FTP enabled, and have the source code of the node. The Node's files are on the right hand pane and the local PC files are on the left hand pane. Clicking on the Project Directory button will return you to the project folder.


On the right are several buttons:




Refreshes the current listing.

 Create a new folder on the controller.

 View the currently selected file in the right hand pane (opens in Notepad).

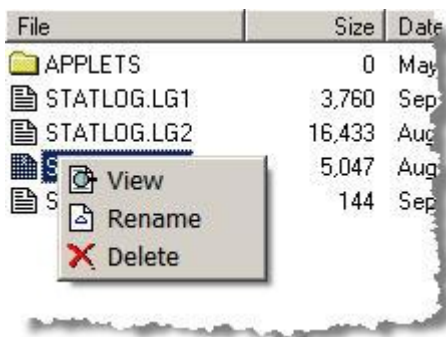
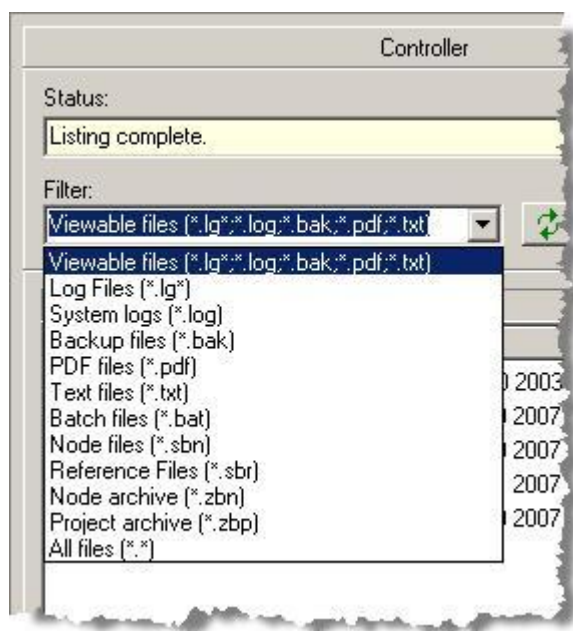
 Rename the currently selected file.

 Delete the currently selected file.

File	Size	Date	Time
APPLETS	0	May 20 2003	00:00
STATLOG.LG1	3,760	Sep 19 2007	15:17
STATLOG.LG2	16,433	Aug 10 2007	11:58
SYSTEM.BAK	5,047	Aug 21 2007	13:35
SYSTEM.LOG	144	Sep 19 2007	15:33

Any given listing will show the folders on the disk and any files with their sizes in bytes and the date and time they were last written to according to the controller's time. Date and time can be handy to see if a log file has been written to recently and in need of uploading for example.

The file Filter specification allows you to pick and choose what files you wish to display. Some disk systems (like this one) get crowded and it is difficult to pick out the files you want.



Right clicking on a file also brings up the speed menu shown here to the left.



When you view a file, ScadaBuilder will look for a file type association (.pdf's are for Acrobat Reader for example) and open the file. If no association is found then the file will be opened in Windows Notepad.

Uploading Files

To upload files, simply select them in the right hand pane and click on the << button. The file will be uploaded to the directory shown in the right had pane. This is handy for uploading your log files (.lg1 or .lg2) as well as the system.log file that can help you with troubleshooting more difficult problems on the system.

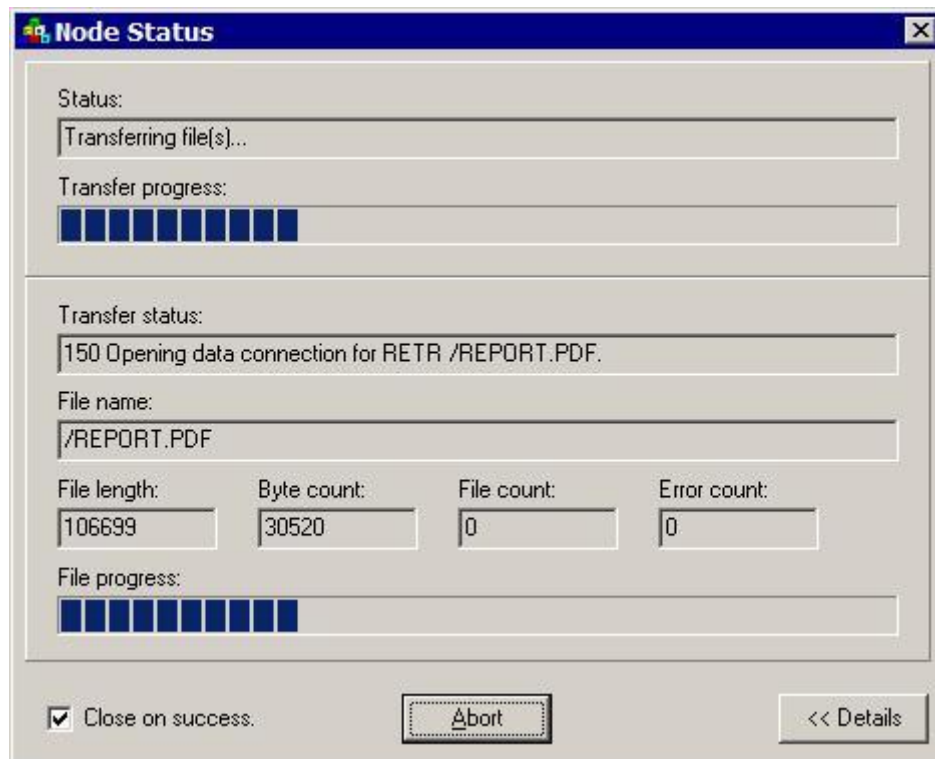
Downloading Files

To download files to the controller, select the file in the left pane and click on the >> button.



The File Transfer Window also supports dragging and dropping files from pane to pane and also from Windows Explorer.

In either case, you will see a progress dialog. You can abort at any time.



If you want less or more details, click on the << Details button.



Send Setup To Host

This feature allows for the remote download of files via Bricknet or Modbus. See **Remote Host File Transfers** (on page 281) for more information.

Report Generator...

This option allows you to output the ScadaBuilder configuration into PDF format for insertion into a customer OEM manual.

Click on the **Tools | Report Generator...** menu

Report Generator

Setup Options

Report setup file:

Filename:

Load... Save...

Title page:

Title: Slave2 Configuration Report Default

Model: EtherLogic LC Default

Revision: Revision 1.0 Date (leave blank for system):

Image file: Browse

Company: Developer:

Address line 1: Phone:

Address line 2: Website:

Generate Close

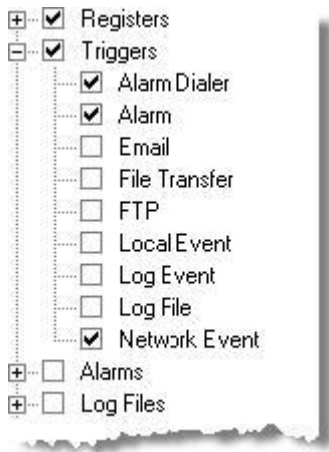
The Data here is for the title page of the report. If you wish to have a watermark image on the front cover, provide the path to the image file in the field above.

Click on the Option tab.

This configures which parts of the ScadaBuilder configuration you want in the report.

Anything checked in here will show up on the report tree. For example if only Registers and Triggers are checked, then all registers will be shown but those that have triggers applied to them will also have the Trigger record configuration in a subordinated section following the Register definition.

- ☒ Registers
- ☒ Triggers
- ☒ Alarms
- ☒ Log Files
- ☒ Dialers
- ☒ Local Events
- ☒ Local Alarms
- ☒ Network Ports
- ☒ Network Sessions
- ☒ Textual User Interface
- ☒ Voice User Interface
- ☒ GPS Setup
- ☒ I/O Scaling
- ☒ I/O Configuration



If your application does not utilize a section (for example, you have no VUI) then uncheck it from the report generator.

If you do not want a subordinated section to show up in a main section, you can expand the tree and uncheck the undesirable components.

You can also select and unselect all components.



A good rule to follow is:

If you don't need it, uncheck it. These reports can get very large.

Go back to the Setup tab. Here you can save your report configuration by clicking the Save button and telling the file dialog where to save the .rpg file.

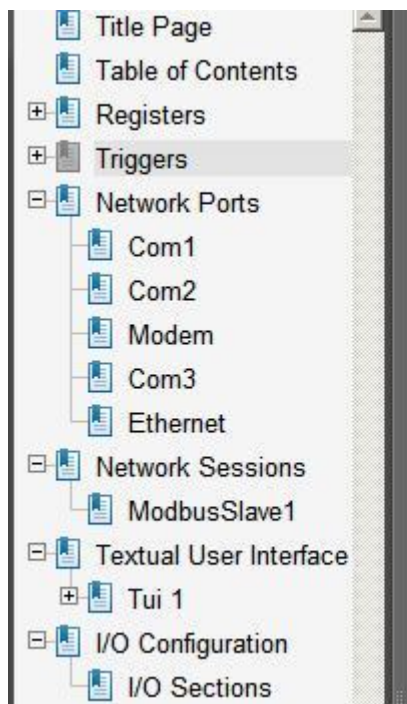
Loading is the reverse process.

Click the generate button and the report should generate and open for you.

You must have Adobe Acrobat reader 4.0 or later.



Your report should look like this with proper headers and page numbers.



Triggers

ArchiveNowTrig

Properties:

Type:	state on
StartReg:	ArchiveNow
BlockSize:	1
Retrig:	0
DebounceOn:	2
DebounceOff:	0

ArchiveNowResetTrig

Properties:

Type:	state on
StartReg:	ArchiveNow
BlockSize:	1
Retrig:	0
DebounceOn:	2
DebounceOff:	0

LogTrigger

Properties:

Type:	timer 2
Source:	Tick

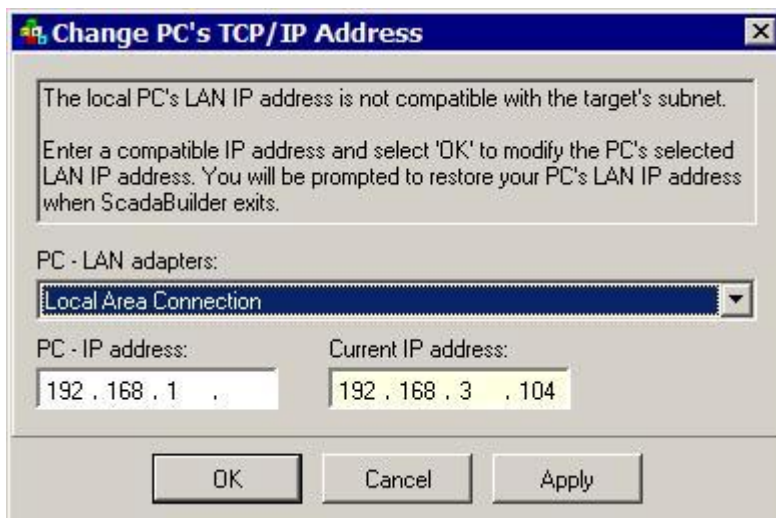
Auto LAN Restore

This option allows ScadaBuilder to temporarily change your IP address on a particular interface on the PC in order to communicate with the controller over TCP/IP and FTP. When ScadaBuilder exits, it will set the PC interface back to the address it was before.

When the option is checked and you attempt to do a Target download operation at any time, FTP is enabled on that node and the download mode is set to FTP then you will get this dialog here.

Enter the last octet (typical configuration). It cannot be the same as the Current IP Address field.

ScadaBuilder will attempt to download over FTP at this point.



Be sure that applications like email clients, FTP clients and Windows Explorer windows looking at network shared drives are all shut down before clicking OK on the above dialog.

This dialog will come up if you try to debug ISaGRAF  over Ethernet and the Auto Lan Restore option is checked.

Restore LAN Address

After the Auto LAN Restore option has had its way with your Local Area Connection interface, you can manually restore the old address (even if it was obtained from a DHCP server).



Your old LAN IP address will be restored when you exit ScadaBuilder.

URL List

This list is really configured in the HTTP Section of ScadaBuilder when you enter permissions. For each permission, you can have a web page associated with it. This URL list provides a quick link to those pages on the currently selected node. See **HTTP Permissions Editor** (on page 44) for more details on permissions and pages.

Export To Ergoview



This utility export all register numbers and names formatted so that Ergoview can generate new instances of Modbus registers for use in Ergoview (VIB Laces) screens. This is the same as essentially exporting a tag database.

The Server loader instrumentation bean will read the file in and generate Modbus interface data.

The time base is how often each of the points gets polled. The Station address is almost always 1.

The ScadaBuilder Hierarchy

ScadaBuilder is a configuration tool with a complex tree and branch structure that allows it to handle large numbers of individual data points (registers) or handle large blocks of data. Knowing the hierarchy of ScadaBuilder and its more esoteric block handling features can save development time and program memory as well. It can also improve program execution time as well as network throughput on telemetry systems.

With ScadaBuilder, there are some basic records covered in the manual here as individual configurations including Registers, Triggers, Local Events, Network Events, Log Events, Alarms, Local Alarms, Dialers, Call Groups, Log Alarms, Network Destinations, Network Sessions, Network Ports, and Log Files.



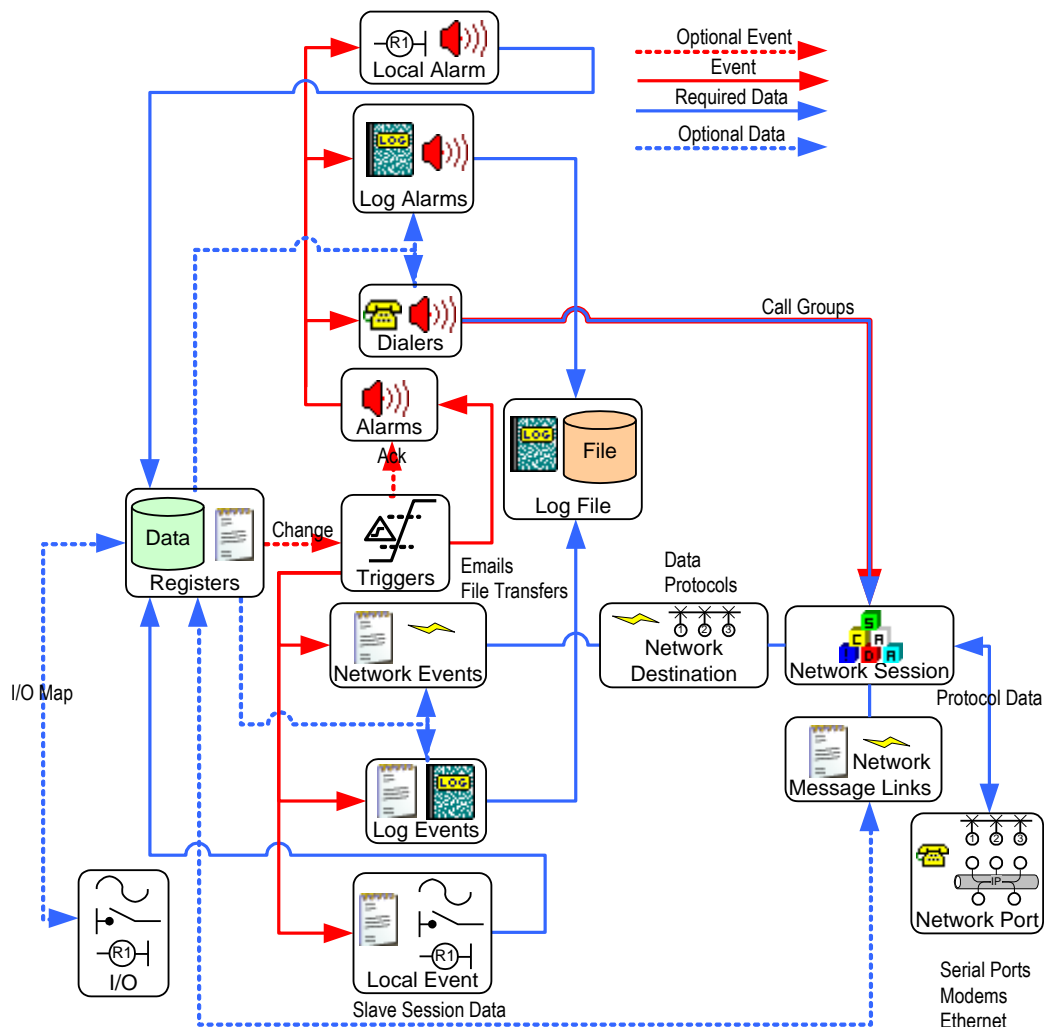
With ScadaBuilder, there is also the Textual User Interface (TUI) that can use some of these configurations directly.

- TUI Fields, TUI Value Lists, and TUI Buffers access Registers in the system
- TUI Alarms for displaying Alarm states
- TUI Buttons for use as a Triggers
- TUI Logs to display Log Files in real time

The Voice User Interface can also access currently unacknowledged and acknowledged Dialer Alarms from any Call Group.

For example, a Network Port is an entity unto itself and can be programmed in many ways. Multiple Network Ports can reside on a unit at any given time. But the real power of ScadaBuilder is that it can use a Network Port for multiple purposes. Practically any number of master-protocols can use a single Network Port and share it nicely.

The same is true for many other records. Records in ScadaBuilder are made to be reusable in many cases to do several different jobs at once. Here is a simplified diagram of the hierarchy of ScadaBuilder's record types:



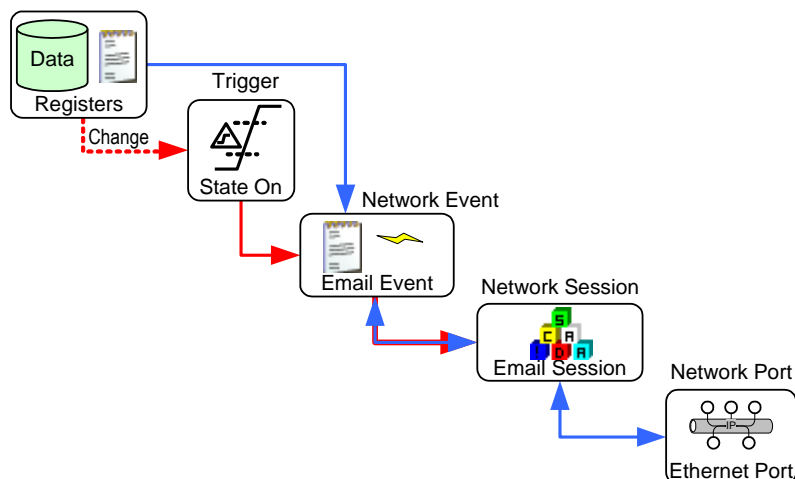
Here, the red lines represent a change of state or triggered events. This is one record or telling another when to do its job. The blue lines represent required or optional data reading from or writing to the register to be used by the activated record.

This looks daunting at first but let's break it down into smaller pieces.

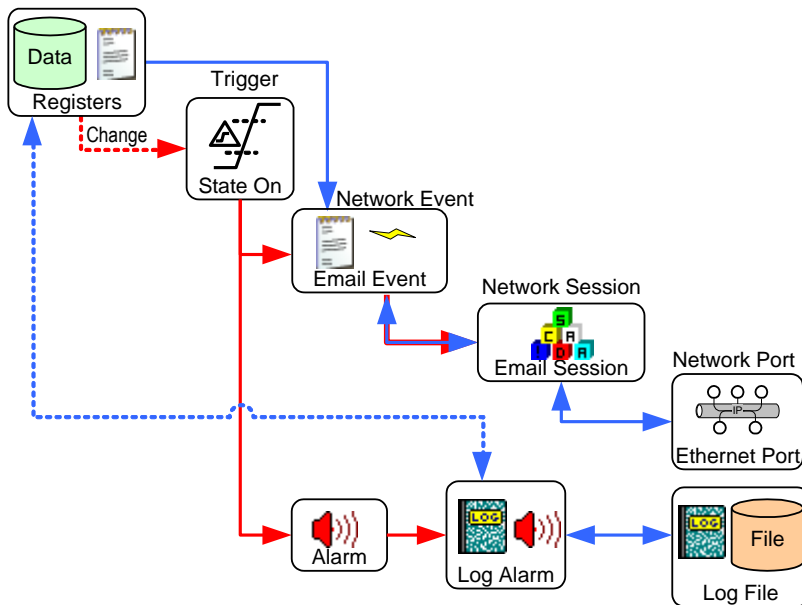
For example, A Trigger (State on) might be used To trigger an Email to be sent out when a boolean register is set to true.

The Email (another form of Network Event) can contain live register data that is inserted when the Email is composed for transmission.

The Networks Session tells where to deliver the Email and what Network Port to use.



Trigger Reuse



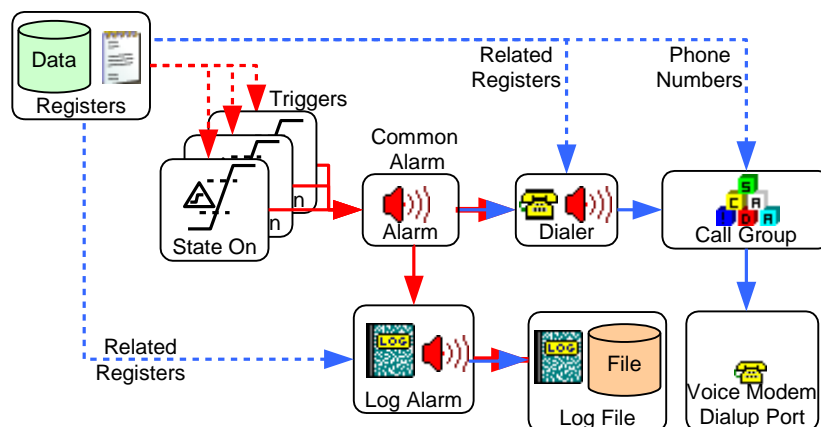
Several Records are involved in this operation but reuse adds an efficient way to utilize configurations that you have already setup. To our previous example, let's add an Alarm, Log Alarm and Log File to the mix so we can Email and Log our Alarm for later review:

Only the three records for the Alarm, Log Alarm and Log File need to be added to complete the task.

Triggers can also be combined to allow an Alarm to be reused for a common alarm configuration...

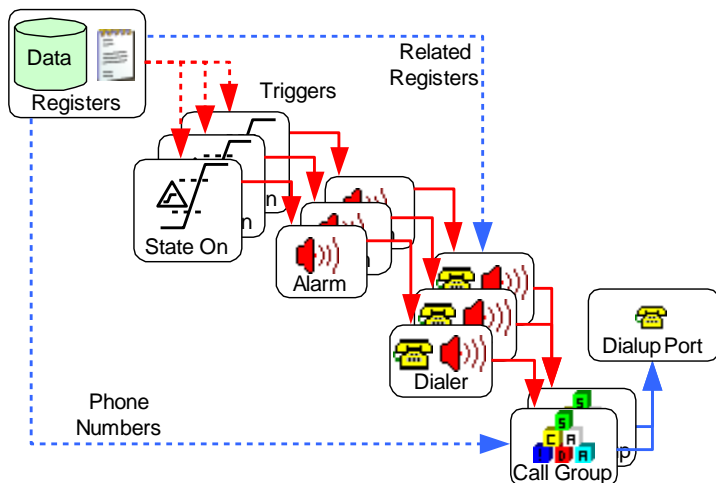
In this case, three different Trigger conditions are applied to one Alarm. This Alarm does two jobs.

It uses a Dialer to call out over a dialup link and annunciate an Alarm and Register values to a user who in turn may acknowledge that Alarm over the phone.



The Alarm Triggers a Log Alarm which can format and record the offending data (or any register or text data you wish) and save it to a Log File on the controller for later retrieval. Any change in the Alarm state will cause a write to the Log File through the Log Alarm.

Multiple Dialers and Call Groups

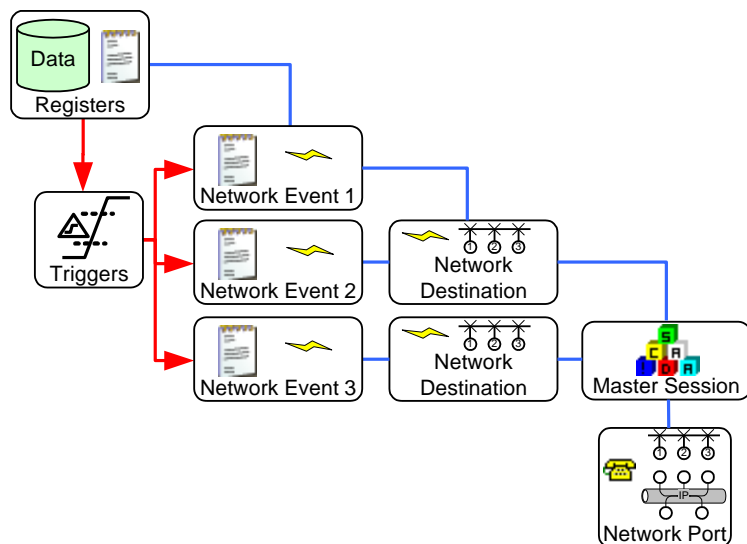


In this case, three Triggers fire three individual Alarms to three Dialers. Two of the Dialers share Call Group 1 and the third Dialer utilizes Call Group 2. Both Call Groups share a single dialup modem Network Port. If either interface is active, then it will take control of the port.

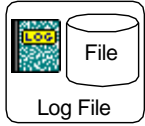
If both are activated at the same time, the first one in the list (first one created) has priority. The Call Groups can have completely different phone number lists so that emergency type alarms may be delivered with one of the Call Groups and maintenance alarms may be delivered with the other.

Using a Trigger To Activate a Polling Cycle

In this case, it is necessary to make a system where there is one Modbus Master and multiple Modbus Slaves. The Master can activate its Network Event list with just one Trigger to start the polling cycle. The Network Events will fire one at a time (because they use the same Network Session) and progress through the Network Destination list and subsequently the Network Event list in the order of their definition.



Using Log Files -- Data and Alarm Logging



See *The ScadaBuilder Hierarchy* (on page 73)

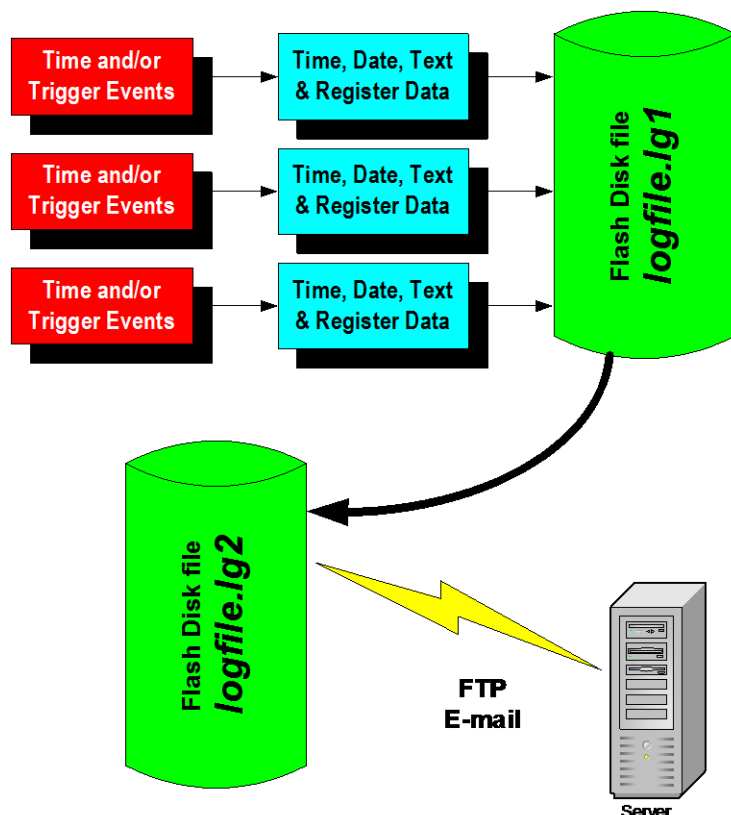
ICL controllers have a (flash) “disk” and a real time clock, enabling them to record data and Alarms with date and time stamps. The logs can be displayed with the built-in TUI MMI, send/retrieve as files using FTP protocol, transferred serially via Modbus or Bricknet, or e-mailed over an Ethernet or serial/modem connection.



Log Grabber - A log file collection program called Log Grabber is available to simplify collecting log files and depositing them in spreadsheet compatible form on the PCs hard drive. Log Grabber can connect to a controller via the Internet, an Ethernet network, or by a dial-up (PPP) telephone connection.

Log files are stored in human-readable ASCII format. ScadaBuilder makes it easy to configure the data logging parameters without programming by simply filling in the blanks in a pair of configuration windows. When data is logged (periodically and/or upon events), the data format, maximum file size, registers, text headers and data delimiters can be set to format the data. Formatting the data as it's recorded makes it's easier to use with standard software tools when the files are retrieved.

ScadaBuilder provides the option of compressing the logged data to increase the effective data logging capacity of the controller. The compression scheme is designed to record the log values at the beginning of each log file, and subsequent entries are recorded only on changing data. Other than available disk space, there is no limit on the number of data logs that can be running simultaneously.



Each ScadaBuilder Log File is implemented using two files. An “lg1” or “lg2” extension is appended to the user selected file name (i.e. logfile.lg1 and logfile.lg2). The controller always writes data to the .lg1 log file. When the file reaches either a specific number of records or file size, the controller checks to see if an lg2 file already exists. If not, the .lg1 file is renamed with an .lg2 extension and logging continues to a new .lg1 file. If a previous .lg2 file exists, the controller can be configured to either overwrite it (overwrite the oldest data) or continue to fill the current lg1 file.

Typically, the .lg2 file is the one that is retrieved by a host system. When the .lg2 file is created, ScadaBuilder can automatically perform an FTP file transfer or e-mail the file to one or more destinations (a “push system”). It can also set a flag to signal a Host system to retrieve the file (a “pull system”). Once the file is transferred or retrieved, it is typically deleted so that when the latest .lg1 file is complete, it can be renamed to an .lg2 extension again.

For more information about transferring files over different media, check out these topics:



- **TOOLS Menu** (on page 64), File Transfer... section.
- **Creating Emails** (on page 370)
- **Creating an FTP Event** (on page 376)
- **File Transferring Over Modbus and Bricknet** (on page 277)

Pinnacle and Later Controllers

The LG1 LG2 mechanism can be overridden to just build a single file by unchecking the File Archive Enable checkbox.

Data can also be logged directly to USB flash drives. Holding the button next to the Status LED will suspend writing to USB flash drives but the data will be written to RAM while the USB flash drive is absent. When the USB flash drive is re-inserted, the data will be written to the USB flash drive as soon as it mounts. If the log file does not exist, a new one will be created.

Files also support headers and footers as well so XML and HTTP files can be supported by putting field terminators in the headers and footer section of the log file.

Files can also be written to subdirectories. Simply specify the subdirectory in the file name. The subdirectories must be created ahead of time from the console or from an FTP client tool or from the ScadaBuilder Tools | File Transfer Tool...

If an extension is specified in the file name and archiving is disabled, that extension will be used verbatim. This is useful for creating .CSV and .XML files.

In This Section

Creating a Log File.....	80
Log Files Reference	82
Using Log Events	91
Using Log Alarms	97

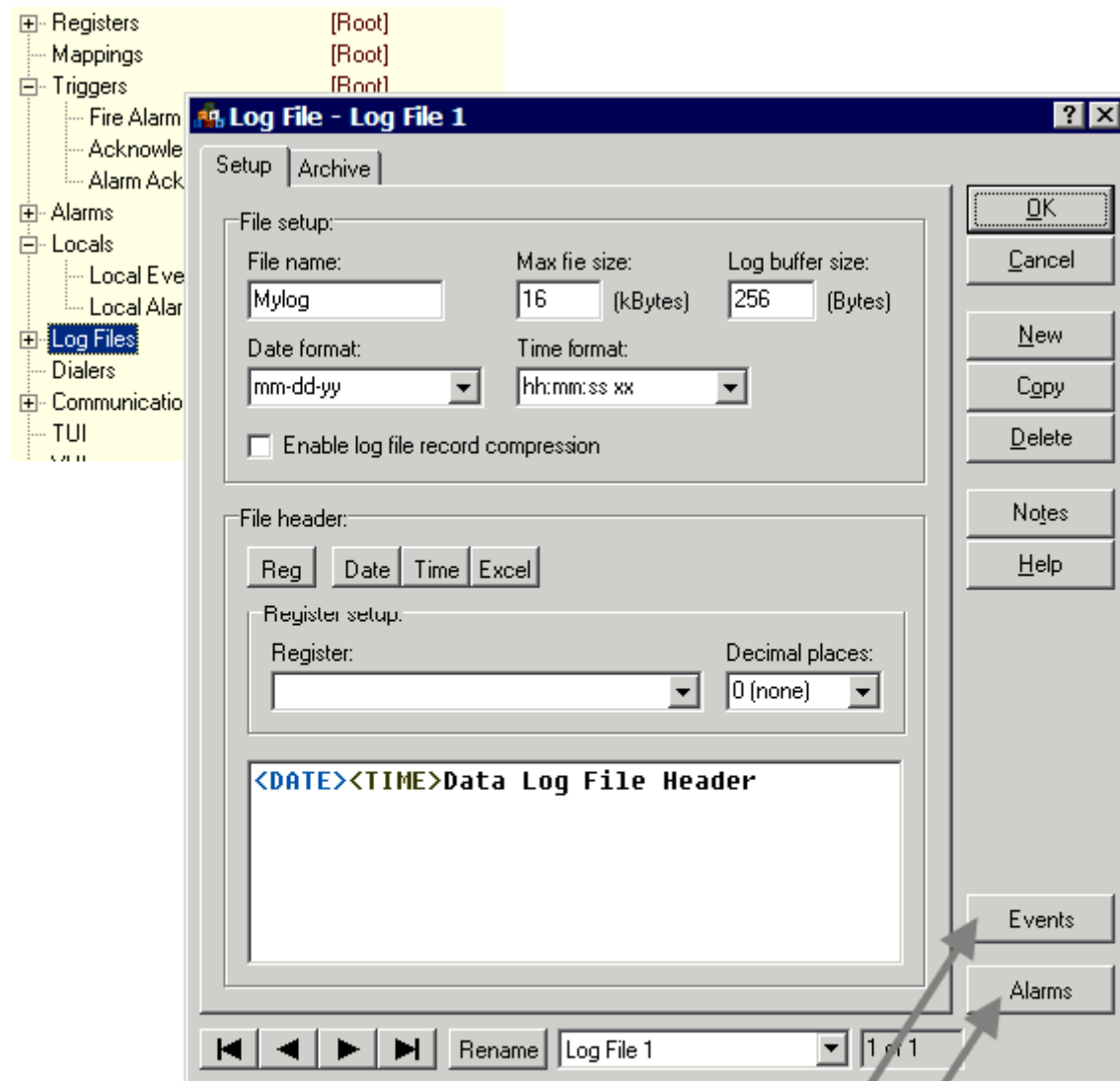
Creating a Log File

To create a Log File, double-click on “Log Files” in the Project Manager (if there are no other log files defined) or click on “NEW” in the log files window. A window will pop up for naming the log. Note that this is NOT the name of the log file itself. You’ll have a chance to specify that later. Accept the default name or enter a new one.

After naming the log, ScadaBuilder displays a log configuration window that defines all of the elements specific to a single log file EXCEPT for the data itself. This includes the:

- file name
- limits on the size of the file
- compression
- and the format of a file header written to the start of each log file

After these basic parameters are defined for the file, the parameters that control the actual log data are specified in separate windows; one for “Events” and a second for “Alarms”. This allows for various types of Triggers and Alarms to record different kinds of data into a single file. For example, you may define a configuration that periodically logs a set of process variables. If an alarm occurs, the alarm data will be recorded interspersed with the process data to simplify analysis of what happened that led to the alarm condition.



After setting up the log file, use these buttons to setup the data to be logged.

See **Using Log Events** (on page 91), and **Using Log Alarms** (on page 97) sections for more details.

The Archive tab governs most of the conditions and controls that have to do with archiving from the .lg1 to lg2 file. The tab has data such as:

- archive triggering
- log status (data ready) flag
- maximum records count
- overwrite options

The screenshot shows the 'Archive' tab of a 'Setup' dialog box. It contains the following elements:

- File archiving:**
 - Max records count:** A text box containing '500' with '(0 = disabled)' to its right.
 - Archive activation:**
 - Trigger:** A dropdown menu with a downward arrow and an 'Add...' button to its right.
 - A text box below the dropdown containing 'Archive_Trigger'.
 - Force archive/new header at startup:** An unchecked checkbox.
 - Disable archive file overwrite:** An unchecked checkbox.
 - Archive indicator map:** A dropdown menu showing 'Archive_Status (5001)'.

For more information on these parameters see *Log Files Reference* (on page 82) section for details.

Log Files Reference

A Log File logs register values and alarm states along with date/time stamps. A Log File can also be displayed on a TUI (Textual User Interface). You specify the base file name of a Log File, but the file extensions are fixed.

Each ScadaBuilder Log File actually corresponds to 2 files stored by the operating system. The extension ".lg1" is used for the file that is currently being updated. The extension ".lg2" is used for the file when it is archived.

You specify the maximum byte size or record count of the ".lg1" file. The file system must allow room for twice the specified byte size. This provides room for the archive file as well (".lg2" extension).

When it is time for logging system to make a new log entry and the ".lg1" file has reached the specified size or count, the file will be archived (renamed with the file extension ".lg2"). The new data will be written to a newly created update file (containing the ".lg1" extension) and then logging will continue as normal. This will always preserve a data history of at least the size or count specified for the Log File. (This 2-file system is used because entries cannot be efficiently removed from the beginning of a file).

The Log File system also provides Compression, Disable Overwrite and Header options that may be used as needed by an application. See the specific help sections for details on these options.

Log File Setup Tab

File Name

Specifies the name (with an optional path) of the file to use for logging. The logging system will concatenate file extensions as needed. The extension ".lg1" is used for the file that is currently being updated. The extension ".lg2" is used for the file when it is archived.

If a log file extension such as ".csv" is specified, and File Archiving enable is unchecked, then the system will not use the naming convention above and will just continue to write to one file with the extension specified.

If File Archive Enable is checked and an extension is specified such as ".csv" then it will write to <Logfile Name>.csv1 then archive to <Logfile Name>.csv2.

Subfolders may be used here by specifying a path in front of the file name ie:

\\Logs\Logfile.<ext>

Max File Size

Sets the maximum size (in Kbytes, 1 Kbyte = 1024 bytes) of the "update" (.lg1) log file. The logging system uses two files, the "update" file with an ".lg1" file extension and the "archive" file, with an ".lg2" file extension. When the update file reaches the maximum size, it is archived and renamed to the .lg2 file. Logging resumes with an empty update .lg1 file. Thus, the file system must have room to accommodate twice the specified maximum size.

Log Buffer Size

Sets the size of the buffer that is used for sending log entries to the file system. Due to the inherent time required by the system to update the file system, buffering is required. Applications that perform many file operations and log large data blocks at a frequent rate may need to increase this buffer. The symptom of this buffer being too small is that log entries (or characters) will be missing from the log file.

A "W-LOG-Log buffer write overrun occurred" error will be posted to the system.log file.

NOTE: Increasing this buffer size uses up memory resources in the system, so it is advisable to keep it as small as possible.

ICL4300, Etherlogic, and ScadaFlex controllers set this parameter in Bytes. Pinnacle and later controllers set this parameter in KBytes.

Date Format

The Date Format specifies the format of any date stamps that are inserted into the log messages (Event or Alarm) created for the log file. The components of the format are:

- m - month digits
- d - day digits
- y - year digits

Time Format

The Time Format specifies the format of any time stamps that are inserted into the log messages (Event or Alarm) created for the log file. The components of the format are:

- h - hour digits
- m - minute digits
- s - seconds digits

- xx - am/pm indicator

Enable Compression

Set this checkbox to enable compression. When compression is enabled, register values that have not changed since the last log entry will not be written to the file. This is useful for conserving disk space on the file system.

If the log file is delimited, then the delimiter character will still be inserted after each compressed entry as a place holder. Any trailing delimiters will not be written to the file. If none of the values have changed since the last log entry (all compressed) then a single "@" character will be inserted for the entry.

The first entry in the log file, either after a system restart or after an archive has occurred, will never be compressed.

Drive

Pinnacle controllers only.

IDE, USB1, USB2, USB3, or USB4 may be specified as the destination drive to store the log file.

Register

Selects the register for which the value will be inserted into the log file header when the "Register" button is clicked.

File Header

The File Header specifies the text that is printed at the top of each log file when it is initially generated or archived. The value of one or more registers along with the date and time can be inserted into header. The header is optional and can be used to label columns of data or insert time/date stamp information. Also available is an Excel time stamp which can be read directly into Microsoft Excel.

Log File Header Data Types.

<i>Registers</i>	Registers of any type may be inserted into the Log Event message area including buffers (messages).
<i>Date / Time / Excel</i>	Date and time format are allowed in standard and Excel recognized format.
<i>Text</i>	Any free form text needed.

Decimal Places

Specifies the number of decimal places that will be printed for a floating point register when inserted into the log file header with the "Register" button.

File Footer

This field is useful for creating file terminators for HTTP and XML files which can be served up directly in a web browser. The footer field is removed and rewritten every entry write so only static text is allowed.

XML example:

Log File - Log File 1

Setup | Archive

File setup:

File name: \WebRoot\LogFile.xml Max file size: 1 MBytes Log buffer size: 100 KBytes

Date format: mm-dd-yy Time format: hh:mm:ss xx Drive: IDE

☐ Enable log file record compression

File Header:

? | Reg | Date | Time | Excel

Register setup:

Register: Decimal places: 0 (none)

<XML>

File Footer:

</XML>

OK
Cancel
New
Copy
Delete
Notes
Help

Events
Alarms

Navigation: [Back] [Forward] [Stop] [Refresh] Rename: Log File 1 1 of 1

HTML example:

Log File - Log File 1

Setup | Archive

File setup:

File name: \\WebRoot\\LogFile.html Max file size: 1 MBytes Log buffer size: 100 KBytes

Date format: mm-dd-yy Time format: hh:mm:ss xx Drive: IDE

☐ Enable log file record compression

File Header:

? Reg Date Time Excel

Register setup:

Register: Decimal places: 0 (none)

File Footer:

<HTML><BODY>

</BODY></HTML>

OK Cancel New Copy Delete Notes Help

Events Alarms

Navigation buttons: [Back] [Forward] [Home] [End] [Rename] Log File 1 1 of 1

Register Button

The "Register" button inserts the register specified in the "Register Setup" box into the log file header. The current value of the register will be printed whenever the header is generated.

Excel Button

Adding this field to a logfile header will mark the time that the header was written (meaning a new LG1 file was started) in Microsoft Excel format. This format consists of days since Jan 1, 1980 and marking the current time of day in a 1/10000th of a day format. This format ports directly into an Excel spreadsheet.

Date Button

The "Date" button inserts the date into the log file header. The date as formatted in the Log File (such as "mm-dd-yy") will be printed whenever the header is generated.

Time Button

The "Time" button inserts the time into the log file header. The time as formatted in the Log File (such as "hh:mm:ss") will be printed whenever the header is generated.

Events Button

The "Events" button allows you to define and edit event log messages that will be printed to the Log File whenever a specified event is activated.

Alarms Button

The "Alarms" button allows you to define and edit alarm log messages that will be printed to the Log File whenever a specified alarm changes state.

Read Only (Setup Tab)

Items with the Read only option are limited to read only access as expected. The Read only flag will also allow the same Access Code to be applied to multiple read only entries.

When setup this way, the system, upon receiving the code, will play all of the read only registers with that code in the order listed. This keeps the user from having to enter multiple codes just to read data.

Non-read only entries can still allow access to the same registers--just through different access codes.

Log File Archive Tab

File Archive Enable

If this box is checked, the normal <ext>1/<ext>2 file extensions are used in the default manner described in the File Name parameter. If no file extension is specified then the standard .LG1 and .LG2 file extensions are used.

If this box is unchecked then any extension specified in the File Name will be used verbatim with no numeric index appended to it. This is useful for creating .CSV, .HTML or .XML files.

File Archive Enable is automatically unchecked if High Speed Log Enable is checked.

Max Records Count

Sets the maximum number of record entries (message logs) that are written to the log file before it is archived. This allows log files with a uniform number of record entries to be generated. Setting the value to '0' will disable the count, causing an archive to be performed only when the **Max File Size** (on page 83) is reached.

Archive Activation Trigger

Allows user to specify a Trigger or list of Triggers that activate a log file archive manually.

Force New Header

Set this checkbox to enable the forcing of a new header into the log file when the system is restarted. If enabled, the system will try to perform an archive when restarted. If the archive can be performed a new "*.lg1" update file is simply generated with a new header file. If the archive cannot be performed (the "*.lg2" archive file already exists and disable overwrite is enabled) then a new header is inserted into the existing "*.lg1" update file.

This feature is useful to re-synchronize log files that do periodic updates based on a time/date stamp referenced in the header. This allows a new reference time to be automatically generated whenever the system is restarted.

Disable Archive File Overwrite

Set this checkbox to disable overwrites of the archive file. When overwrites are enabled, whenever it is time to do an archive the "*.lg2" archive file is automatically overwritten by the "*.lg1" update file.

When overwrites are disabled, the "*.lg2" archive file is preserved if it already exists and it is time to perform another archive. In this case, log entries are continued to be added to the "*.lg1" update file (as long as the max file size has not been reached) and the archive remains pending. As soon as the "*.lg2" is removed from the system (such as by a file transfer event) then the "*.lg1" file will be immediately archived to "*.lg2".

Archive Indicator Map

Maps a Boolean register to the log file archive indicator. Whenever a log file archive is performed this register will be set to a value of '1'. It is up to the control program to reset the value in this register. This indicator is useful in creating a trigger that activates the processing or sending of the log file.

High Speed Log Enabled

This check box is enabled when the **Enable I/O Scan Sync Trigger** (on page 36) in the **Node Settings - General Tab** (on page 34) check box is checked.

High Speed Logging may be used to specify a number of samples or a sample time. Once triggered, the I/O scan will run at the specified rate in **I/O Scan Sync Rate** (on page 36). The data will be logged in a time stamped file specified by the **File Name** (on page 83) and suffixed with a date stamp down to 1 second. This name indicates the start time of the sample.

Log Cycle Start Map

Boolean register when set to TRUE by the control program or telemetry, this flag starts the High Speed Logging operation for the specified duration. The register is set to FALSE when the logging is done to indicate that the duration has completed its cycle.

Log Duration (Sec)

High Speed Logging only. Duration of a High Speed Logging operation. When triggered, data will be written to the log file in **I/O Scan Sync Rate** (on page 36) mS multiplied by the **I/O Scan Cycle Multiplier** (on page 89) parameter. This is the Log File sample rate.

This parameter works in cooperation with the **Log Trigger Count Max** (on page 89). Whichever parameter is reached first will cause the High Speed Logging to stop at that point and reset the **Log Cycle Start Map** (on page 88) boolean to be reset.

Log Trigger Count Max

This parameter limits the number of entries to write in a High Speed Logging operation. This parameter works in cooperation with the **Log Duration (Sec)** (on page 88) parameter to limit the number of entries in the file.

Whichever parameter is reached first will cause the High Speed Logging to stop at that point and reset the **Log Cycle Start Map** (on page 88) boolean to FALSE.

I/O Scan Cycle Multiplier

High Speed Logging only. Final acquisition speed of a High Speed Logging operation. When triggered, data will be written to the log file in **I/O Scan Sync Rate** (on page 36) mS multiplied by the I/O Scan Multiplier parameter. This results in the Log File sample rate. Each High Speed Log file can have its own I/O Scan Multiplier.

For Example:

If the I/O Scan Sync Rate is 10 mS and the I/O Scan Multiplier parameter is 3, the data will be logged every 30 mS.

Log File Name Map

Map a Message register here to get the name of the file created when a High Speed Log operation is concluded. This file name is written after the Log Cycle Start Map Boolean is set to false meaning the logging operation is done.

The file will follow the <Log File Name>_YYYY_MM_DD_HH_MM_SS.[<ext>] format. If no extension is specified, then LG1 will be used.

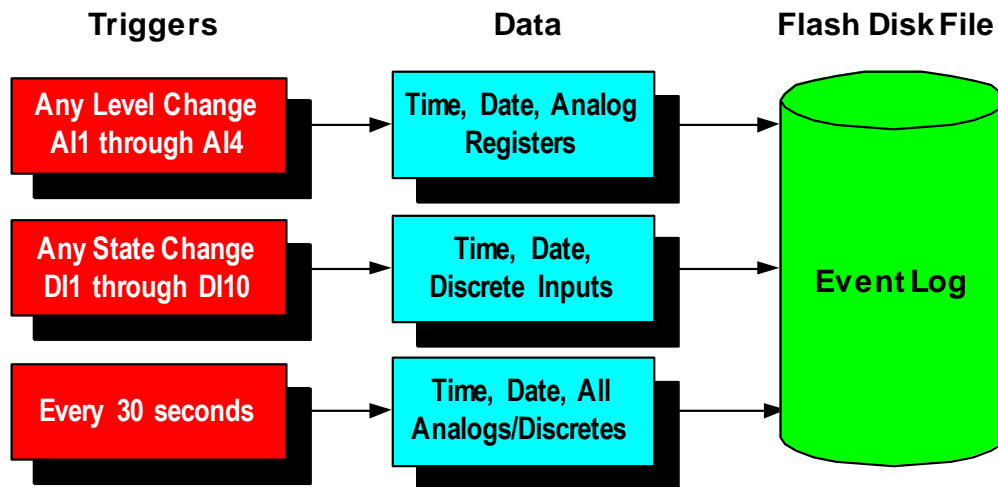
Each file will be named according to the start time and date of the acquisition.

Using Log Events



See *The ScadaBuilder Hierarchy* (on page 73).

Once a Log File has been created, you can define how data will be stored in the file and under what conditions it will be written. This is called a “Log Event”.



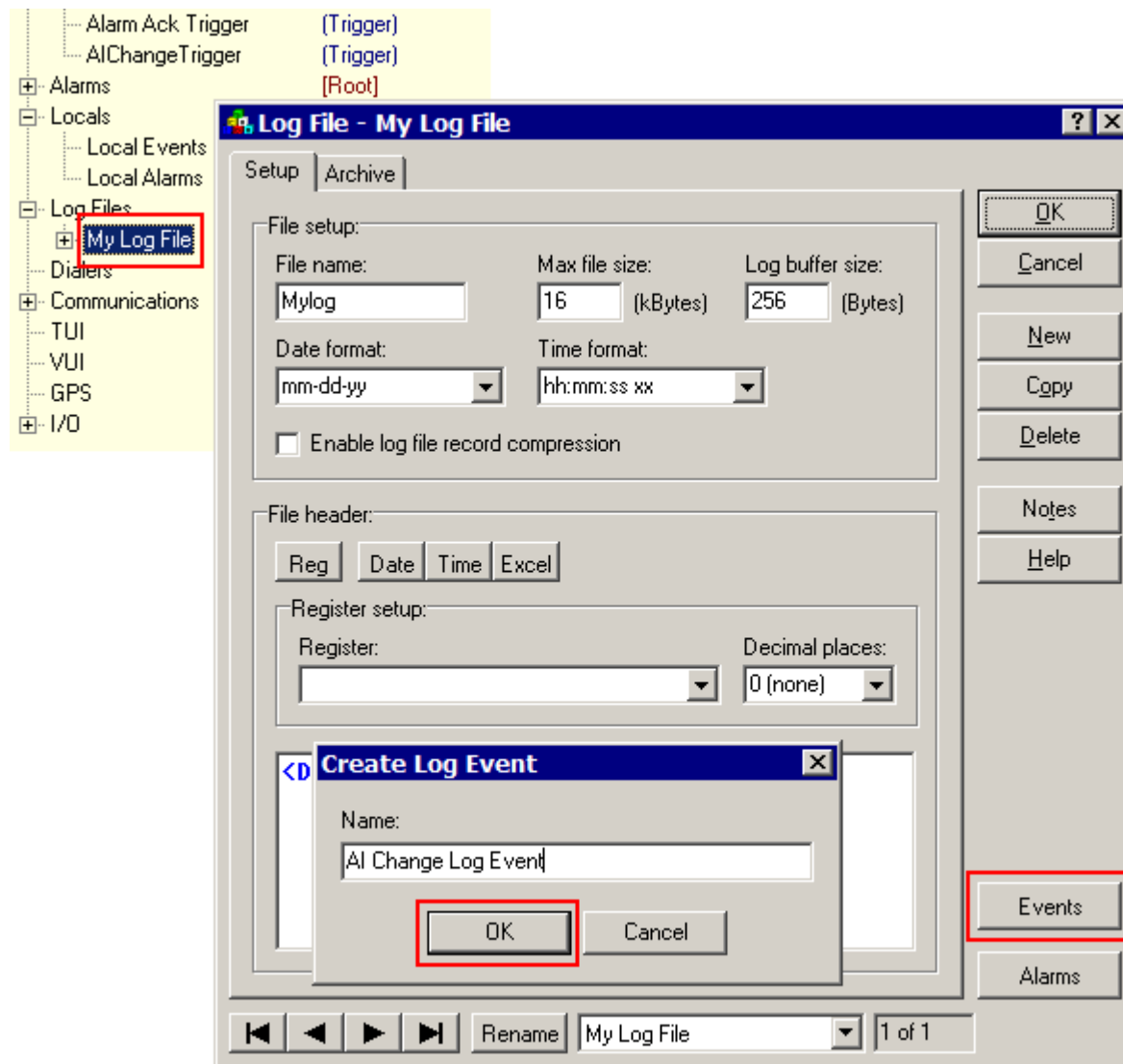
Log Events are recorded to the controller's flash disk based on user defined events known as Triggers. Triggers are used throughout ScadaBuilder. They can be based on the change of a numeric value, a boolean state and time. Multiple triggers can be used to write a single data record, or separate Triggers can write separate types of data records to the same file.

In This Section

Creating a Log Event.....	91
Log Events Reference.....	94

Creating a Log Event

To create a new log event, open a log file (double click on it in the tree view if you like) and select the Events Button. You will be prompted for a new Log Event name. Enter it here and click OK.



Log

Log activation:

Trigger

AIChangeTrigger
One Hour Trigger

Add

Log delimiter:

Delimiter:

Comma

Startup:

☐ Trigger on startup

Log message:

Reg Date Time Excel Tick Elap Count Delim

Register setup:

Register:

Decimal places:

0 (none)

Block size (constant / register map):

1

<COUNT><DATE><TIME><REG:AI1><REG:AI2><REG:AI3>
<REG:AI4>

Define your Trigger or Triggers

Data is written a Log File based on one or more Triggers caused by a change in an analog or boolean value, an analog level going above or below a setpoint, or a time interval.

To specify a Trigger, select a previously defined Trigger in the “Trigger” window or add a [New] one, then click on the “Add” button. More than one Trigger can be used for a Log Event. For example, the data log file could be updated whenever one of the analog levels changes and once per hour.

Enter any data you wish in the Log Message frame.



If you would like to enter multiple lines in a single Log Event, then just hit Enter at the end of line and go to the next. A new line will be added when the Event is triggered.

Log Events Reference

A Log Event defines the content and format of a record to be stored to a log file, as well as the condition(s) that cause the record to be written. The Log Event record can contain date and time stamps, register values and literal text.

Trigger

The Trigger selection list allows you to select a Trigger that is to be added to the list of Triggers that cause the Log Event to be written to the log file. Select the desired Trigger, then click the Add button.

Trigger Add Button

The Add button causes the selected Trigger to be added to the list of Triggers that activate the Log Event.

Log Delimiter

The Log Delimiter is a character that will be placed after each log item that has been inserted into the log message. This only applies to token items (like <REG>, <DATE>, <TIME>) and NOT plain text that has been entered into the message.

Trigger at Startup

If "Trigger on startup" is enabled, the Log Event will automatically be activated when the application starts.

Log Message

The log message specifies the text that is printed to the associated Log File whenever the Log Event is activated. The value of one or more registers along with the date and time can be inserted into log message.

Log File Entry Data Types.

Registers	Registers of any type may be inserted into the Log Event message area including buffers (messages).
Date / Time / Excel	Date and time format are allowed in standard and Excel recognized format.
Tick	This is the number of milliseconds since startup. This timer free runs.
Elap	This is the number of milliseconds since the last archive.
Count	This is the number of records since last archive and makes for a good way to count records
Delim	Inserts one character of the current delimiter (space, tab or comma) - whatever is configured in the delimiter selection.

Register

Selects the register for which the value will be inserted into the log message when the "Register" button is clicked.

Block Size

Allow user to configure a start register and a block size for the Logfile record. To use this feature, the indexes or "network addresses" of the registers must be in contiguous order. The Blocksize parameter does not allow for gaps in register numbering.

The user may also map a register to control the Blocksize from runtime to runtime. The register should be retained or initialized to the Blocksize value. It is only read once at startup.

Decimal Places

Specifies the number of decimal places that will be printed for a floating point register when inserted into the log message with the "Register" button.

Register Button

The "Register" button inserts the register specified in the "Register Setup" box into the log message. The current value of the register will be printed whenever the log entry is activated.

Date Button

The "Date" button inserts the date into the log message. The date as formatted in the associated Logfile (such as "mm-dd-yy") will be printed whenever the log entry is activated.

Time Button

The "Time" button inserts the time into the log message. The time as formatted in the associated Log File (such as "hh:mm:ss") will be printed whenever the log entry is activated.

Excel Button

Adding this field to a log entry will mark the time that the Log Event was written in Microsoft Excel format. This format consists of days since Jan 1, 1980 and marking the current time of day in a 1/10000th of a day format. This format ports directly into an Excel spreadsheet.

Tick Button

Inserting this field into a Log Entry will show how long it has been since the last archive operation as the entries are gathered in the file.

Elap Button

This is the number of milliseconds elapsed since the last record was recorded. This is handy for the testing of triggering and logging delays in higher speed systems. It can also be used in conjunction with the header date and time to know the current time of the entry. Since it does not access the Real Time Clock, the reading of the time is much faster and more accurate.

Count Button

Inserts the count into a Log Entry or Log Alarm. This is the number of records that have been recorded since the last archive. This count is a retained variable so it will not reset through a normal power down of the controller.

Using Log Alarms

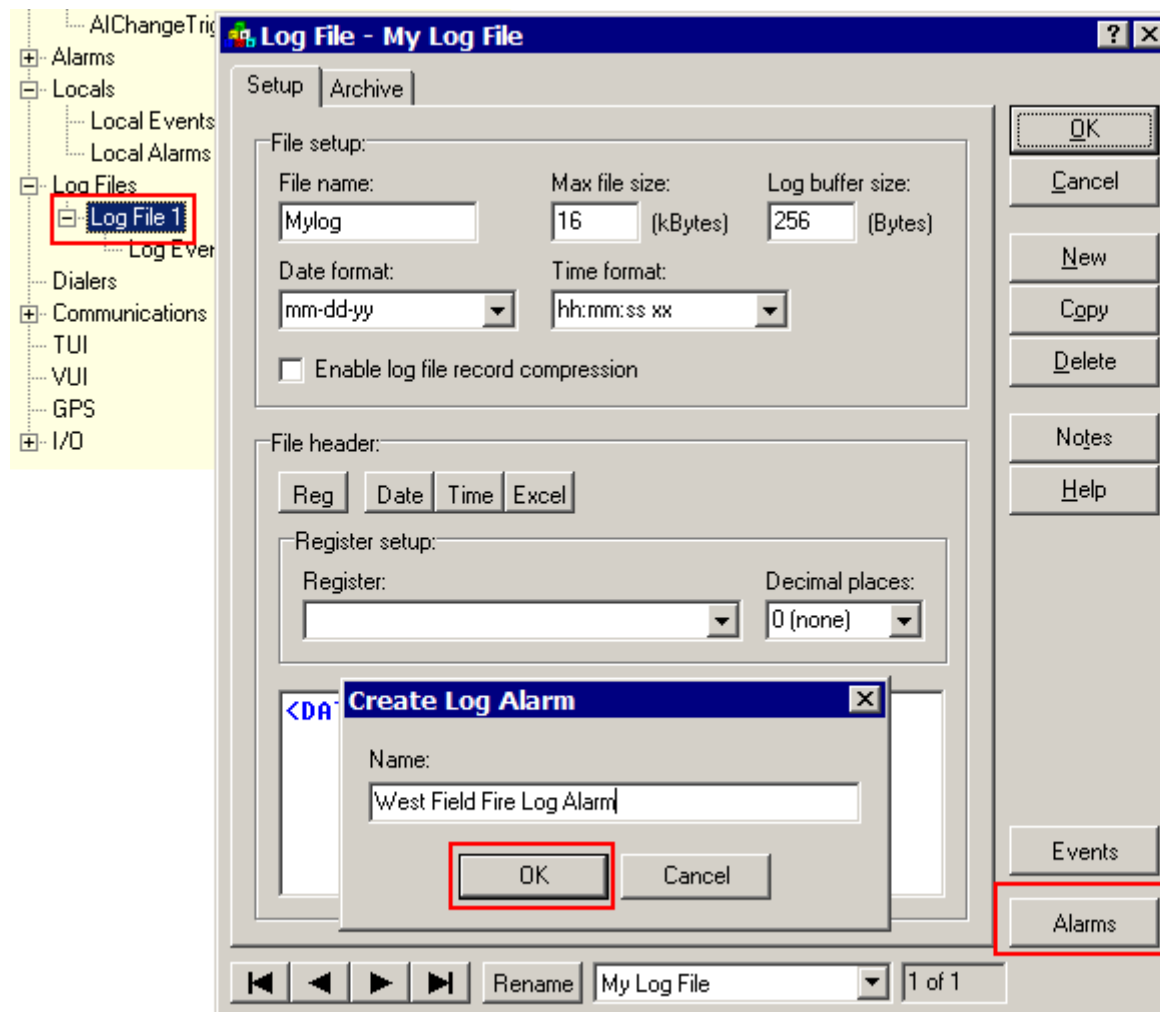
Log AlarmsFILE2257

In This Section

Creating a Log Alarm.....	97
Log Alarms Reference.....	98

Creating a Log Alarm

To create a Log Alarm, click on “Alarms” in the Log File window (if there are no other log events defined) or click on “NEW” in the Log Alarm window. A dialogue window will pop up for naming the Log Alarm. Accept the default name or enter a new one.



After creating and/or selecting your alarm, click the Add button to add it to the list.

More than one Alarm record may be assigned to the Log Alarm. If this is done, the Alarm and the state of the Alarm are recorded where ever the Alarm and State fields are applied in the Log message section.

Log activation:

Alarm

[New]
West Field Fire Alarm

Add

Unlike Log Event records, usually there is a one for one relationship between a single Alarm and the alarm data put into the Log Alarm that shows the change of state for that Alarm. This does not have to be the case though, since any data in the controller can be included in the alarm data record. This means that any process information that might be useful in interpreting the cause of the Alarm can be associated with the Alarm itself.

Log Options

Log activation:

Alarm

West Field Fire Alarm

Add

Log delimiter:

Delimiter:

Comma

Log message:

Reg Date Time Excel Tick Elap Count Alarm State Delim

Register setup:

Register:

Temperature (1)

Decimal places:

0 (none)

Block size (constant / register map):

1

<DATE><TIME><ALARM><STATE><REG:Temperature>

Log Alarms Reference

A Log Alarm defines the content and format of alarm records to be stored to a log file as well as the associated Alarm(s) that cause the record to be written. The Log Alarm record can contain date and time stamps, register values, literal text, alarm names and alarm states.

Alarm

The Alarm selection allows you to choose an Alarm that is to be added to the list of Alarms that cause the Log Alarm to be activated. Each time an Alarm on the list changes state, the Log Alarm is activated, causing a record to be written to the log file.

To add an Alarm to the list, select the desired Alarm, then click the Add button.

Alarm Add Button

The Add button causes the selected Alarm to be added to the list of Alarms to the log.

Log Delimiter

The Log Delimiter is a character that will be placed after each log item that has been inserted into the log message. This only applies to token items (like <REG>, <DATE>, <TIME>) and NOT plain text that has been entered into the message.

Log Message

The log message specifies the text that is printed to the associated Log File whenever the Log Alarm is activated. The value of one or more registers along with the date, time, alarm name and alarm state can be inserted into log message.

Log File Entry Data Types.

Registers	Registers of any type may be inserted into the Log Event message area including buffers (messages).
Date / Time / Excel	Date and time format are allowed in standard and Excel recognize format.
Tick	This is the number of milliseconds since startup. This timer free runs.
Elap	This is the number of milliseconds since the last archive.
Count	This is the number of records since last archive and makes for a good way to count records (this value is retained through a power cycle).
Delim	Inserts one character of the current delimiter (space, tab or comma) - whatever is configured in the delimiter selection.
Alarm	Inserts the name of the Alarm that has activated the Log Alarm.
State	Inserts the state of the Alarm as configured on the Options Tab in the current Log Alarm.

Register

Selects the register for which the value will be inserted into the log message when the "Register" button is clicked.

Block Size

Allow user to configure a start register and a block size for the Log file record. To use this feature, the indexes or "network addresses" of the registers must be in contiguous order. The Block Size parameter does not allow for gaps in register numbering.

The user may also map a register to control the blocksize from runtime to runtime. The register should be retained or initialized to the blocksize value. It is only read once at startup.

Decimal Places

Specifies the number of digits after the decimal point that will be printed for a floating point register when inserted into the log message with the "Register" button.

Register Button

The "Register" button inserts the register specified in the "Register Setup" box into the log message. The current value of the register will be printed whenever the log entry is activated.

Date Button

The "Date" button inserts the date into the log message. The date as formatted in the associated Log File dialog (such as "mm-dd-yy") and will be printed whenever the log entry is activated.

Time Button

The "Time" button inserts the time into the log message. The time as formatted in the associated Log File dialog (such as "hh:mm:ss") will be printed whenever the log entry is activated.

Excel Button

Added this field to a Log Alarm will mark the time that the entry was written in Microsoft Excel format. This format consists of days since Jan 1, 1980 and marking the current time of day in a 1/10000th of a day format. This format ports directly into an Excel spreadsheet.

Tick Button

Inserting this field into a Log Alarm will show how long it has been since the last archive operation as the entries are gathered in the file.

Elap Button

This is the number of milliseconds elapsed since the last record was recorded. This is handy for the testing of triggering and logging delays in higher speed systems. It can also be used in conjunction with the header date and time to know the current time of the entry. Since it does not access the Real Time Clock, the reading of the time is much faster and more accurate.

Count Button

Inserts the count into a Log Entry or Log Alarm. This is the number of records that have been recorded since the last archive. This count is a retained variable so it will not reset through a normal power down of the controller.

Alarm Button

The "Alarm" button inserts the Alarm name into the log message. The Alarm name as configured in the Alarm Log Name Field in the associated Alarm(s) will be printed whenever the log entry is activated.

State Button

The "State" button inserts the alarm state into the log message. The alarm states as configured on the "Options" tab of the Log Alarm will be printed whenever the log entry is activated. The default text for each state is described below:

- "Active" - Alarm condition has occurred and has not been acknowledged.
- "Ack" - Alarm has been acknowledged and alarm condition still exists.
- "Idle" - Alarm condition is not active and any previous alarm condition has been acknowledged.

Unacknowledged Text

Whenever the log entry is activated, this is the text that will be inserted into the Alarm <State> field if the state is "unacknowledged".

Acknowledged Text

Whenever the log entry is activated, this is the text that will be inserted into the Alarm <State> field if the state is "acknowledged".

Idle Text

Whenever the log entry is activated, this is the text that will be inserted into the Alarm <State> field if the state is "idle".

103

SECTION VIII

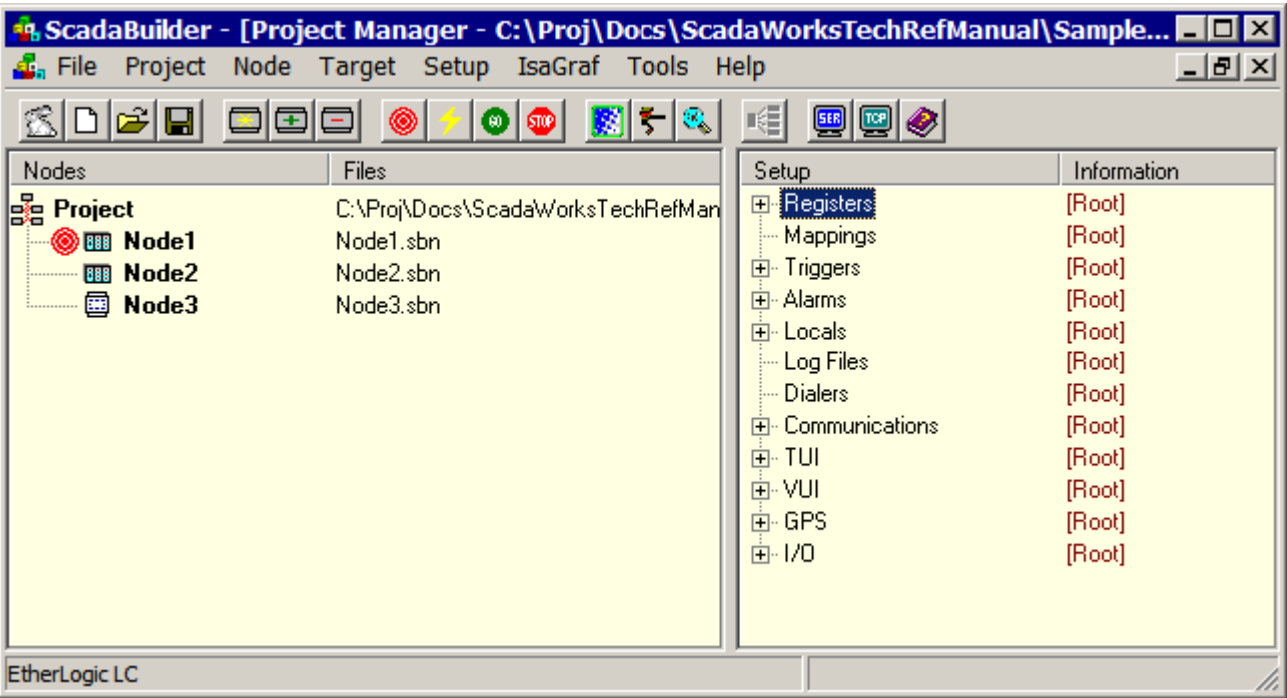
Registers



See: *The ScadaBuilder Hierarchy* (on page 73)

Registers are memory locations for measurements, calculations, and data to be shared with other controllers and computers. In ScadaBuilder, there are four types of registers; Booleans for single bits, 32-bit Integers, 32-bit Reals (Floating Point values) and Messages (also known as buffers, Messages or strings). ICL controllers support up to 65,535 registers. Any of these can be configured as “Volatile” (sets back to specific value or zero on power-on/restart), or “Non-volatile” (contents are preserved through power failures and program restarts).

Register creation and maintenance is a function on the ScadaBuilder “tree”, accessed by clicking on “Registers” as highlighted below:



Double-clicking on “Registers” or on a register type will bring up a new window with the tag names, addresses and comments for the registers. You can choose the register type using the selection box at the bottom center of the windowManaging Boolean, Integer and Real Register FieldsManaging Boolean, Integer and Real Register FieldsManaging Boolean, Integer and Real Register FieldsManaging Boolean, Integer and Real Register Fields

In This Section

Boolean, Integers and Real Register Windows.....	104
Managing and Editing Registers	105
Retained (Non-Volatile) Registers and Initial Values	113
Using Registers in Your Program.....	114
ISaGRAF 3 Register Tools	116
Registers Reference	118

Boolean, Integers and Real Register Windows

The Registers windows are identical for Booleans, Integers and Real numbers. The top portion of these windows has an entry area to create new register definitions, while the lower portion displays the currently defined registers including their indexes (network addresses), initial values, retained flags and attributes. The attributes can be used to control the display of individual registers in various sections of ScadaBuilder making it possible to shorten the length of register selection lists. This feature makes it easier to work with programs that have a large number of registers where only a portion are used in specific areas of ScadaBuilder.

Register Allocation - Integers

Register names:

Prefix: Enum: Template file: Enum: Suffix:

Sample output: Index: 1 Count: 1 ☐ Retain

☒ I/O ☒ TUI ☒ Comm stats (CS) ☒ Network events (NE) ☒ Remote scale (RS)

Name	Index	Value	Rtn	I/O	TUI	CS	NF	RS	Comments
COMM_STAT_48	1048		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
COMM_STAT_49	1049		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
COMM_STAT_50	1050		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
CSTAT_1	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Comm Stat - Transmits
CSTAT_2	1002		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Comm Stat - Receives
CSTAT_3	1003		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Comm Stat - Timeout Errors
CSTAT_4	1004		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Comm Stat - CRC Errors
CSTAT_11	1011		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
CSTAT_12	1012		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Navigation buttons: [Back] [Forward] [Home] [End] [Rename] [Integers] [2 of 4] ☒ Show attribute columns

To make the registers window look less cluttered, you can uncheck the “Show attributes columns” box.

Register Allocation - Integers

Register names:

Prefix:

Enum:

Template file:

Enum:

Suffix:

Sample output:

Index:

Count:

☐ Ret

☒ I/O

☒ TUI

☒ Comm stats (CS)

☒ Network events (NE)

☒ Remote scale (RS)

Name	Index	Value	Rtn	I/O	TUI	CS	NE	RS	Comments
COMM_STAT_48	1048		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
COMM_STAT_49	1049		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
COMM_STAT_50	1050		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
CSTAT_1	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Comm Stat - Transmits
CSTAT_2	1002		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Comm Stat - Receives
CSTAT_3	1003		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Comm Stat - Timeout Errors
CSTAT_4	1004		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Comm Stat - CRC Errors
CSTAT_11	1011		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
CSTAT_12	1012		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

⏮

⏪

⏩

⏭

Rename

Integers

2 of 4

☒ Show attribute columns

Uncheck this box to hide the display attribute column

Unchecking this box leaves the Names, Indexes (network address), Initial Values, Retained flags (makes registers nonvolatile) and Comment text.

!

Be careful. The windows for managing Boolean, Integer and Real registers look nearly identical, so it's possible to accidentally have the wrong one selected! Be sure that you have chosen the right one before adding a register.

Managing Boolean, Integer and Real Register FieldsManaging Boolean, Integer and Real Register FieldsManaging Boolean, Integer and Real Register FieldsManaging Boolean, Integer and Real Register FieldsManaging Boolean, Integer and Real Register Fields

Managing and Editing Registers

Once a register has been created, it appears in a list in the lower portion of the Register window. The register list may be sorted by any field to simplify finding a register or reviewing the settings of the registers. With a single click, you can sort the register list by name, index (address), initial value, attribute or comment.

Register Allocation - Integers

Register names: Prefix: Enum: Template file: Enum: Suffix:

Sample output: Index: Count: 1 1

☒ I/O ☒ TUI ☒ Comm stats (CS) ☒ Network events (NE) ☒ Remote scale (RS)

Name	Index	Value	Rtn	I/O	TUI	CS	NE	RS	Comments
COMM_STAT_48	1048		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
COMM_STAT_49	1049		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
COMM_STAT_50	1050		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
CSTAT_1	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Comm Stat - Transmits
CSTAT_2	1002		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Comm Stat - Receives
CSTAT_3	1003		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Comm Stat - Timeout Errors
CSTAT_4	1004		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Comm Stat - CRC Errors
CSTAT_11	1011		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
CSTAT_12	1012		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Rename: Integers 2 of 4 ☒ Show attribute columns

This list is assorted in ascending order using the Name field as indicated by the small arrow in the header bar.

Sorting Registers

To sort the register list, click on the heading bar above the field that you want to sort by. Each time you click on the header bar, the list will be alternately sorted in ascending or descending order. An arrow in the header bar indicates the current sort field and direction.

Editing Register (Tag) Names

When editing register tag names, remember that these names may be any string up to 32 characters long containing the letters A through Z and the numbers 0 through 9, as well as the '_' (underscore) character. The name must start with A through Z and must be unique among all data types.

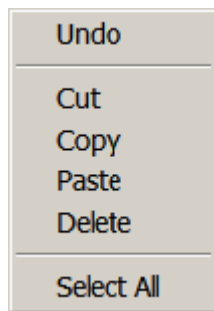
The ScadaBuilder Register editor will tell you if you have named a register the same as another in the system as soon as you leave the edit mode or "Add" a new register.

Right-clicking on the name displays a new selection window that allows the register to be moved up or down in the list. This only changes the displayed position of the register, not its network address. Note that the “Control” and “Up” and “Down” arrow keys can also be used for moving registers up or down in the list.

This can be handy if you wish to renumber a list of registers in particular order after the register has been moved in the list.

Move Up	Ctrl+Up
Move Down	Ctrl+Down
Delete	Del
Edit Selected	Enter
Access Reg...	

The name of an existing register can be “opened” for editing by double-clicking on the name. The name text will turn blue while it is being edited.



Right-clicking on the name opens an editing selection window to cut, copy and paste the text of the register names.

You can also use the standard Windows shortcut keys for these functions: (Ctrl-C for copy, Ctrl-V for Paste, Ctrl-X for Delete, etc.). The register name will be qualified once you move off the field or if you hit enter.

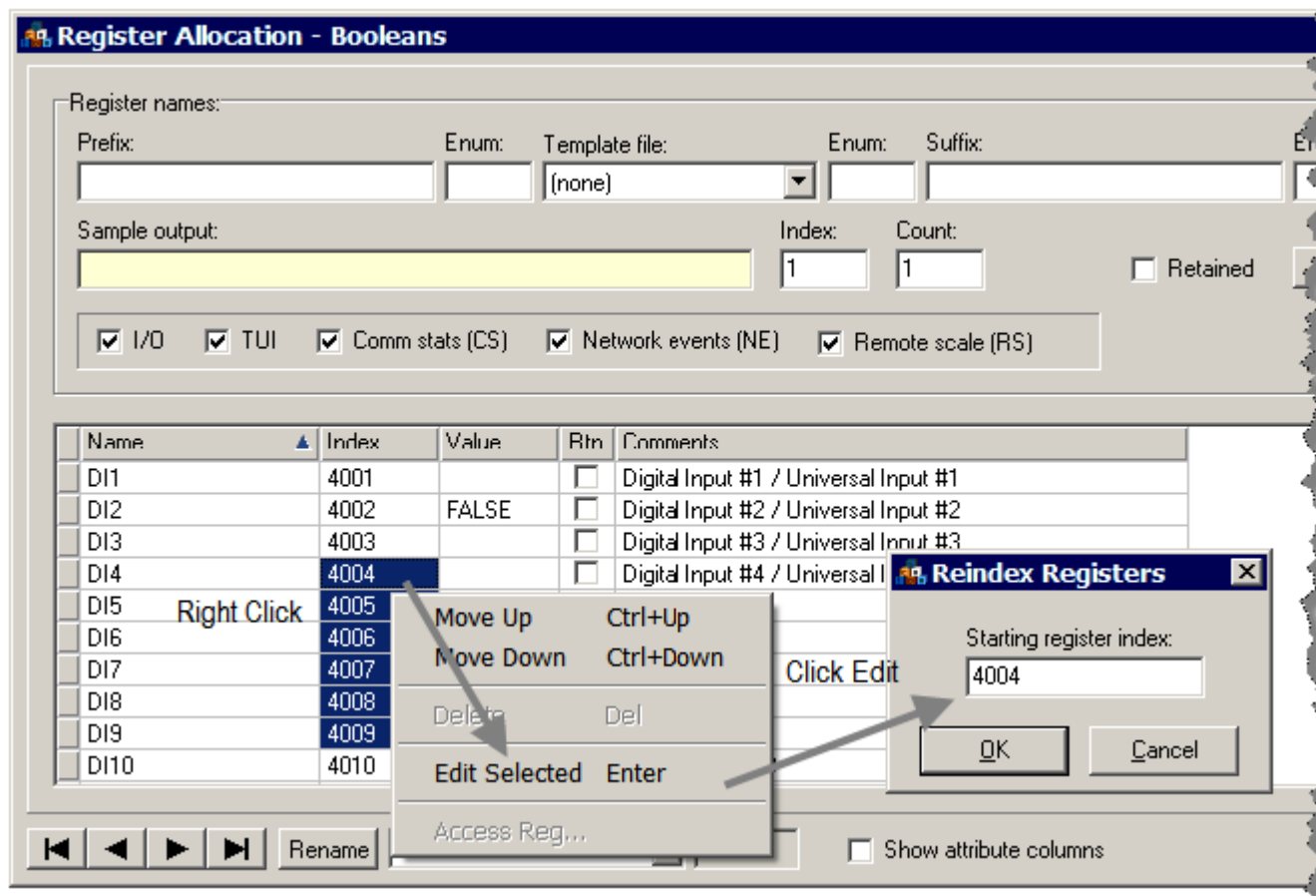
Editing Register Indexes (addresses)

Register indexes may be any decimal number between 0 and 65,535. Each register's index must be unique across all register types except for an index value of 0. An index value of 0 may only be used by ISaGRAF internal variables that will NOT be accessed by ScadaBuilder functions.

Indexes can be moved and renumbered individually or in blocks. Single click on an index to select it; double-click on it to edit it.

To select a block of indexes, click on the individual indexes while holding down the “Ctrl” key, or if the registers are sequential, click on the first index, then hold down the “Shift” key while clicking on the last index. You can also “click and drag” your mouse to select multiple registers.

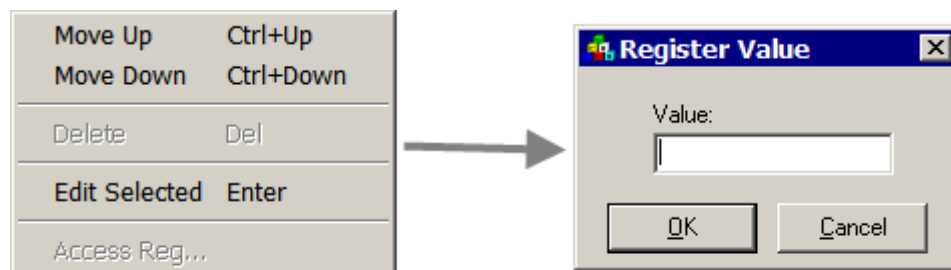
With the index or block of indexes selected, right-click to display a selection menu with functions to move the block up or down in the register list or “Edit Selected” to renumber the indexes. If the later function is chosen, a new window will be displayed to enter the starting index value for renumbering. The first selected index will be changed to this value, then increment it for the next value, etc.



Editing Register Initial Values

The initial value of registers can be set individually or as a block. Single click on a register value to select it; double-click to edit it.

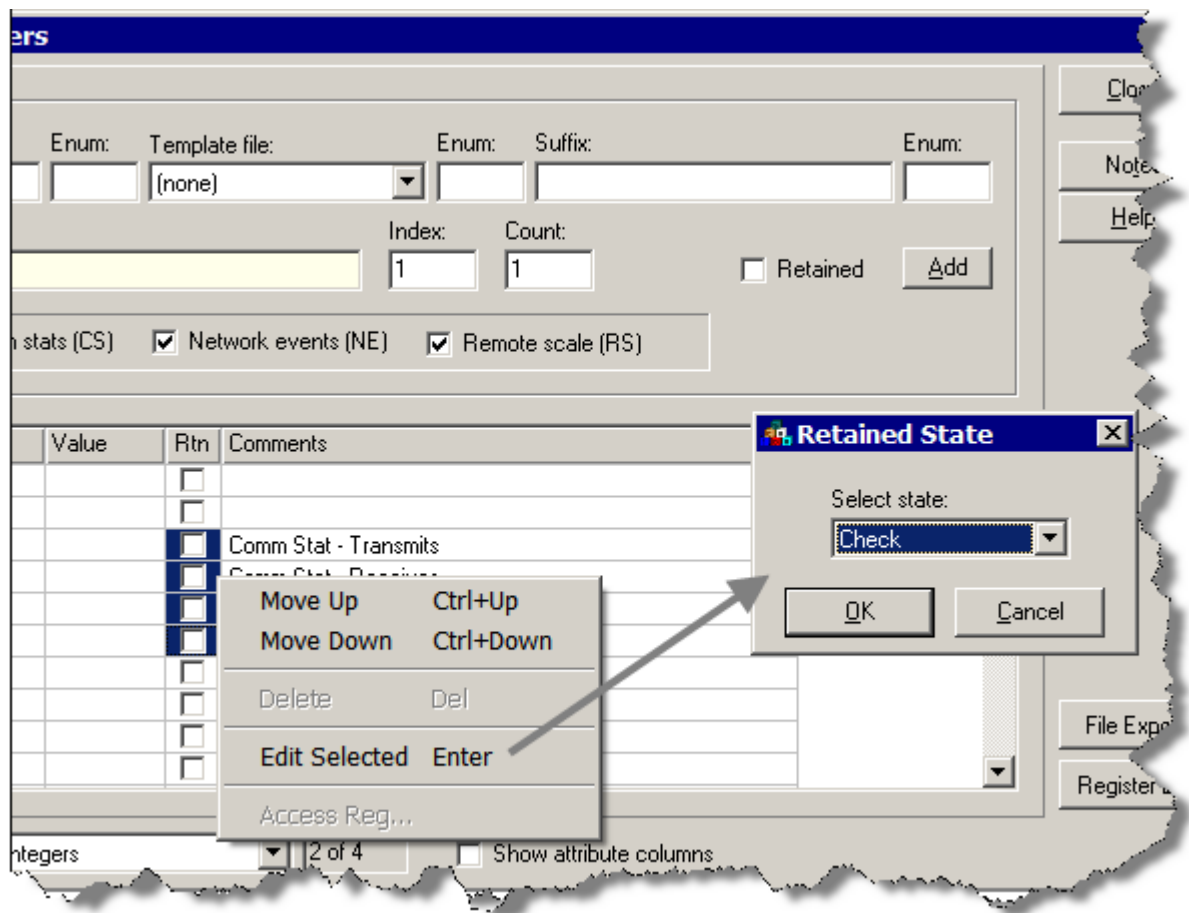
To select a block of initial values, click on individual values while holding down the “Ctrl” key, or if the registers are sequential, click on the first value, then hold down the “Shift” key while clicking on the last value. You can also “click and drag” your mouse to select multiple registers.



With the register value or block of values selected, right-click to display a selection menu with functions to move the block up or down in the register list or “Edit Selected” to change all of the selected values at once. If the later function is chosen, a new window will be displayed to enter the new value for all of the selected registers.

Editing Retained Variable Attributes

Any register may be designated as “retained” so that its value is preserved through a reset or power failure. Registers can be moved and have their retained attribute changed individually or in blocks. Click on a single box to check or uncheck it. To select a block of attribute boxes, click on the individual boxes while holding down the “Ctrl” key, or for sequential registers, click on the first attribute box, then hold down the “Shift” key while clicking on the last attribute box.

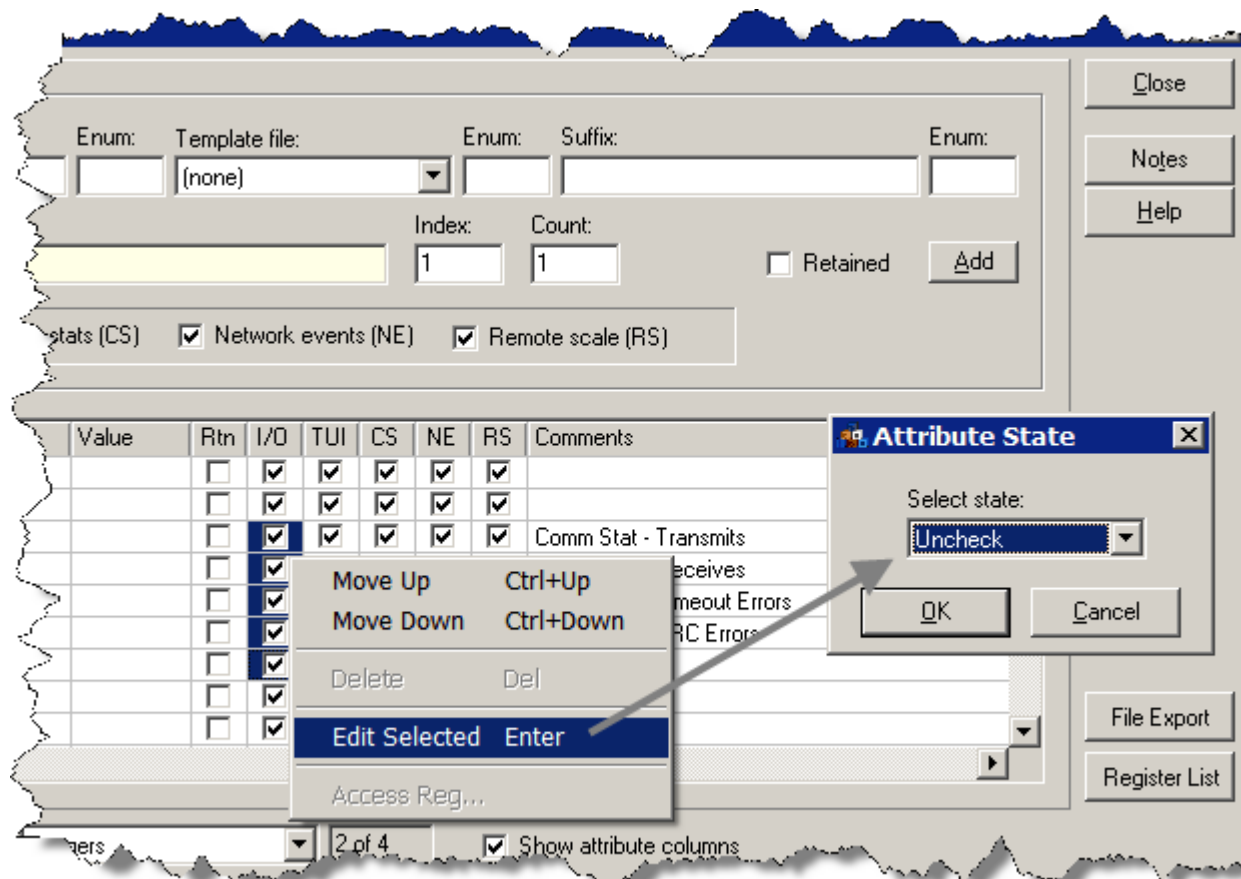


With the attribute box or block of boxes selected, right-click to display a selection menu with functions to move the block up or down in the register list or “Edit Selected” to check or uncheck the boxes all at once. If the later function is chosen, a new window will be displayed to enter the desired retained state (checked or unchecked).

Editing Register Display List Attributes

The display list attributes of registers may be changed individually or in a group to select where registers are included in lists within ScadaBuilder.

Registers can be moved and have their display list attributes changed individually or in blocks. Click on a single box to check or uncheck it. To select a block of attribute boxes, click on the individual boxes while holding down the “Ctrl” key, or if the registers are sequential, click on the first attribute box, then hold down the “Shift” key while clicking on the last attribute box.

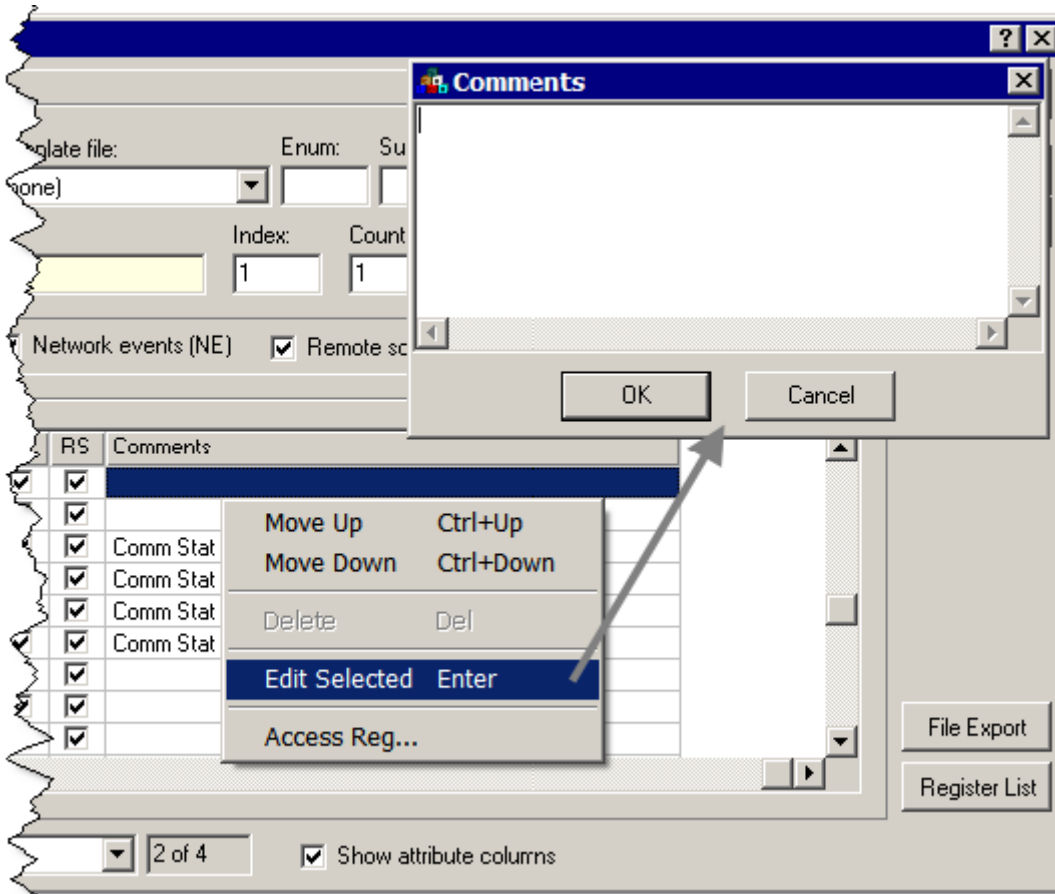


With the attribute box or block of boxes selected, right-click to display a selection menu with functions to move the block up or down in the register list or “Edit Selected” to check or uncheck the boxes all at once. If the latter function is chosen, a new window will be displayed to enter the desired attribute state (checked or unchecked).

Editing Register Comments

Register comments can be changed individually or as a block. Single click on a register value to select it; double-click edit it.

To select a block of register comments, click on the individual values while holding down the “Ctrl” key, or if the registers are sequential, click on the first value, and then hold down the “Shift” key while clicking on the last value in the block.



With the register comment or block of comments selected, right-click to display a selection menu with functions to move the block up or down in the register list or “Edit Selected” to change all of the selected comments at once. If the later function is chosen, a new window will be displayed to enter the new comment text for all of the selected registers.

Message Register Window

The Registers window for Messages (strings) is slightly different from the windows for Boolean, Integer and Real numbers. Message registers have an extra parameter for setting the maximum size of the message, but have no display list attribute parameters.

Template file: Enum: Suffix: Enum:

Index: 1 Count: 1 Size: 12 ☐ Retained

☒ Network events (NE) ☒ Remote scale (RS)

Size	Rtn	Comments
80	<input type="checkbox"/>	
80	<input type="checkbox"/>	
80	<input type="checkbox"/>	
20	<input type="checkbox"/>	
20	<input type="checkbox"/>	
20	<input type="checkbox"/>	
20	<input type="checkbox"/>	
20	<input type="checkbox"/>	
20	<input type="checkbox"/>	
20	<input type="checkbox"/>	
20	<input type="checkbox"/>	
12	<input type="checkbox"/>	

Size Parameter

Format File Export Register List

4 of 4

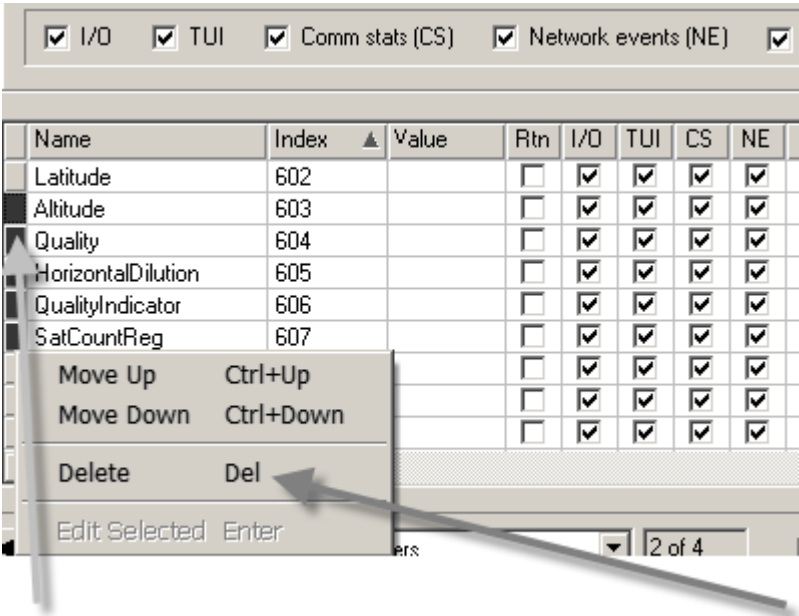
Message Size

The size specifies the maximum number of memory bytes that will be reserved for the message. Not allowing enough memory will cause long messages to be truncated, but allocating excessively large messages will waste RAM resources. Since messages are frequently used to store telephone number strings, the default message size is 12 bytes which works well for telephone numbers with area codes and dialing prefixes.


Editing "in place" in the Registers window

All fields in the Registers window may be edit without right click. Simply click on the "cell" you wish to edit to select it then slowly click again. This places the cell into edit mode and the system will allow you to enter any valid characters you wish such as comments, initial values or registers names. The values entered will be qualified if you hit enter or click on another field. If you wish to cancel editing of a value, simply hit escape.

Deleting Registers



If the register is in use by ScadaBuilder, it will ask you if you want to delete records associated with the register.



If the register is used in ISaGRAF, you will not know it until the next time you do a Make or try to download to the controller.

You can always cancel a deletion if the registers are being used in ScadaBuilder.

Click on the small bar (row selector) and you can optionally drag the mouse to select more than one register or hold the shift key and click a second row selector then right click to get the speed menu shown above.

Click on the Delete or hit the DEL key to remove the registers.

Retained (Non-Volatile) Registers and Initial Values

For Pinnacle and later controllers see *Retained Register Drop Down List* (on page 124).

For Etherlogic, ScadaFlex Plus, and ICL-4300 controllers, Retained variables have a specific interaction with the initial value given to the register. Retained variables are stored in a logically separate area in memory that is battery backed and scanned at startup for coherency. Coherency is checked with a Cyclic Redundancy Check (CRC) at startup. If the CRC does not match what the application is expecting at startup, then all initial values defined for retained variable will be stored to those registers. If the CRC is what the application is expecting, then the retained variables keep their last known value.

During development, the CRC can be changed quite often causing initial values to be stored; the list below defines the possible causes for resetting the retained database:

- Adding or deleting a retained variable
- Adding or deleting a program module
- Renumbering retained variables



It is highly recommended that the programmer enter initial values that will make the program functional during the development period. This can keep the resetting of setpoints and timer variables to a minimum during a testing cycle.

Using Registers in Your Program

Every ScadaBuilder dialog where registers are used utilizes the same interface--a Drop Down Tree. This is done for consistency and versatility.

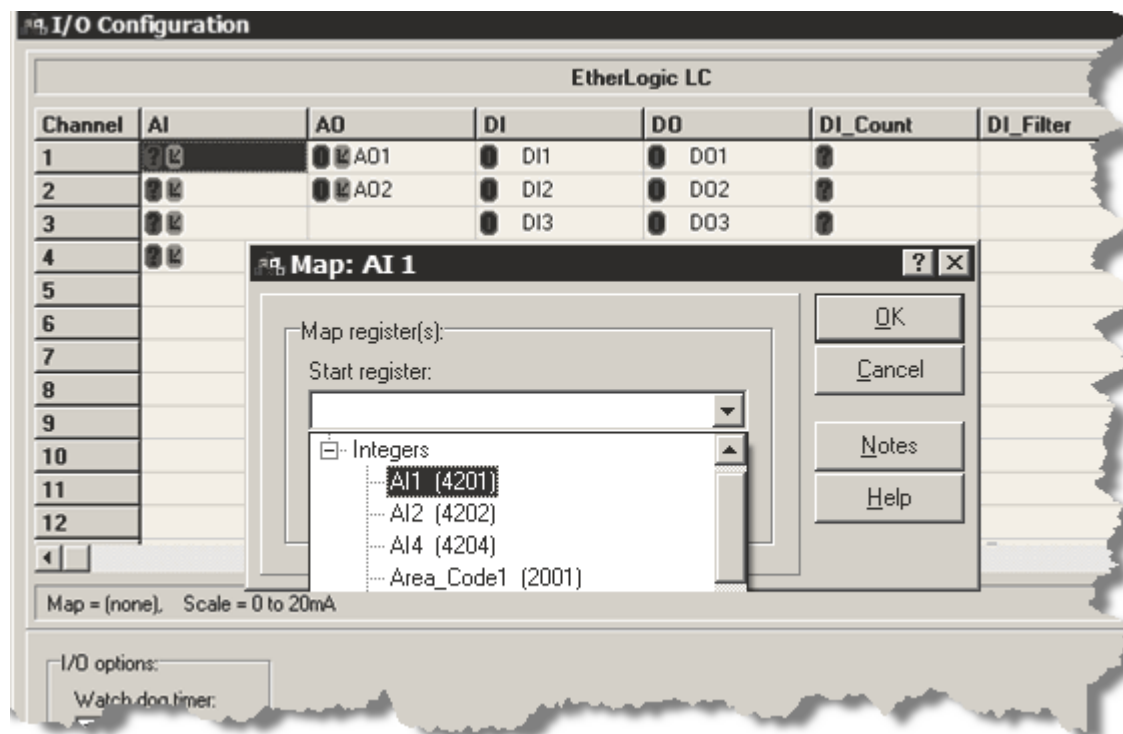
For example, configuration of an I/O point to a register (typically referred to as "mapping" in the ScadaBuilder I/O Configuration tool will look the same as mapping a register in the "Mappings" section of ScadaBuilder. For example, mapping an Analog Input channel, simply:

expand the I/O section under Setup in ScadaBuilder

double click on Configuration

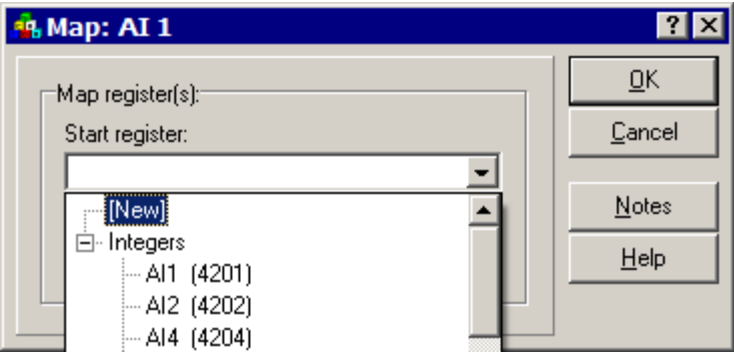
double click on AI channel 1 in the grid

click on the start register drop down and expand the Integers section

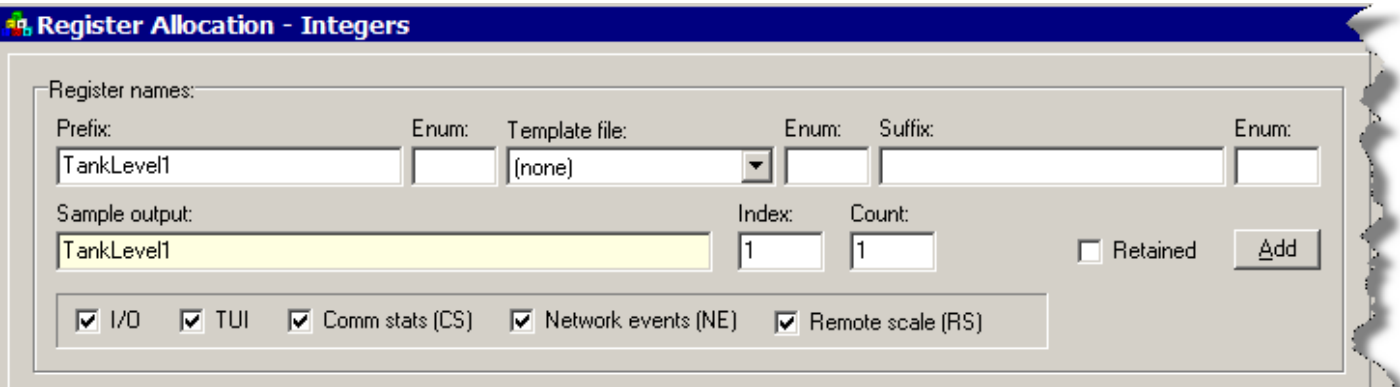


Simply select the register you wish to map to the I/O channel and you are done.

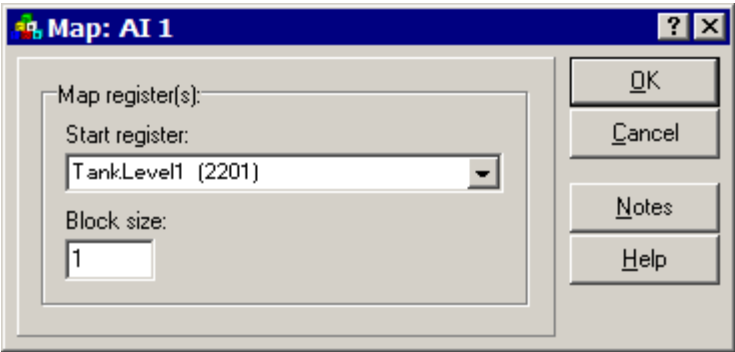
Conversely you can select [New] and create a new register (or block of registers) in the ScadaBuilder Register Editor



Then use the new register to map your I/O channel:













Click Add then Close and you should see:





Notice that the Register Index is also displayed in the drop down as well as the register name.

Click OK and your I/O point is mapped:

Channel	AI	AD
1	 TankLevel1	
2		
3		
4		
5		

This same [New] "Create on the Fly" option can be utilized wherever Registers are used in the ScadaBuilder software package.

ISaGRAF 3 Register Tools



Used for Etherlogic, ScadaFlex Plus and ICL-4300 controllers only. Pinnacle and later controllers cannot and should not use the ISaGRAF 5 import and export tools.



Deleting, renumbering or renaming registers in the ISaGRAF dictionary is not a recommended practice. It is best to do these operations from the ScadaBuilder Registers interface.

ISaGRAF has its own set of tools for editing and managing registers. These tools operate on the same database as the ScadaBuilder tools, although we recommend that you NOT use the ISaGRAF tools for most functions. Besides not having the same capabilities as the ScadaBuilder tools, it is possible to cause database synchronization problems that are not possible when configuring within ScadaBuilder. The exception is the import/export and cross reference tools that are useful, safe, and not duplicated by ScadaBuilder.

ISaGRAF has other data types (Timers, Function Block Instances, and Defined Words) that are used for control programs but NOT used by ScadaBuilder. The Sort and Cross Reference tools are primarily used for writing logic programs and are not described here.

For more information on ISaGRAF programming, please refer to the ISaGRAF Workbench Manual 3.40
http://www.iclinks.com/public_ftp/DocRelease/ISaGRAFWorkbench/v3.40/ISaGRAFWorkbench3.40.pdf

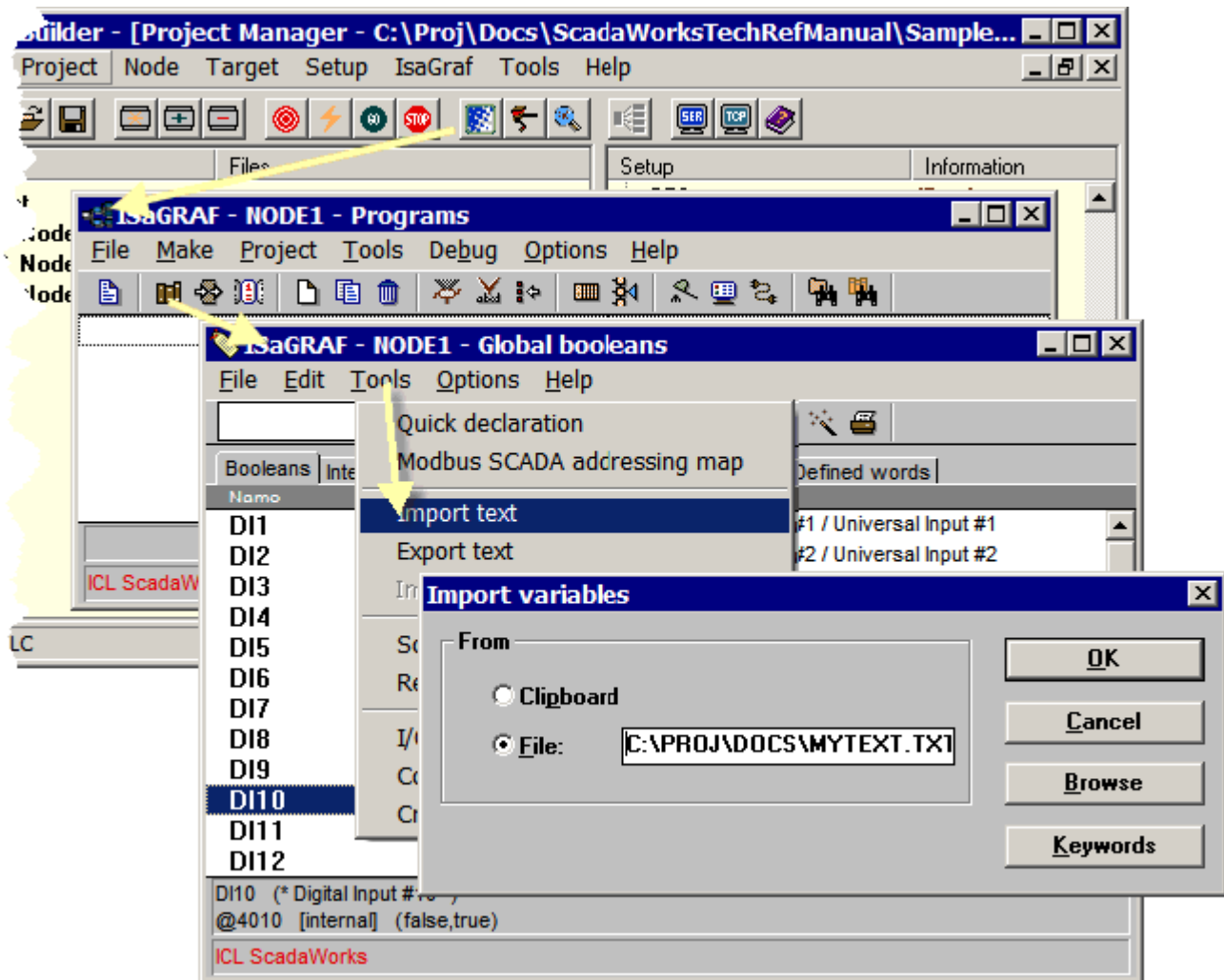
For Pinnacle and later controllers, please refer to the *ISaGRAF 5.20 Manual*
http://www.iclinks.com/public_ftp/DocRelease/ISaGRAFWorkbench/v5.20/WorkbenchV5.20.pdf

Import and Export Text

ISaGRAF provides a means of importing and exporting register information from a file or the Windows clipboard. The import/export format is compatible with most spreadsheets and database applications.

Importing Text

Open the ISaGRAF Programs window and click on the dictionary icon. Select the **Tools | Import Text** menu.



The Import Variables window includes selection buttons for importing from the Windows clipboard or a file, and a browse function to assist in locating the file.

The format of data to be imported resembles a standard ASCII spreadsheet file. It must have a header that describes the field's names on the first line, followed by the new register information, one line per register.

The field separators can be any of the spreadsheet standards such as comma, space, tab, etc. A typical text file to add two registers looks like this (using tab separators):

Name	Address	Attribute	Format
AI5	16#4205	Internal	Integer
AI6	16#4206	Internal	Integer

Note that the register address has format information in front of the register numbers. More detailed information of file formatting is included in the *ISaGRAF User Manual*

http://www.iclinks.com/public_ftp/DocRelease/ISaGRAFWorkbench/v3.40/ISaGRAFWorkbench3.40.pdf.

Exporting Text

The text export function is very similar to the text import function. You can export the entire list of a variable type, or select specific ones. The field separator character is also selectable. The resulting text file is identical to the input example above.

Registers Reference

Registers are simply storage locations for information. Registers are allocated in groups, or "banks." Each register bank has an associated type, and registers in the bank may be used to store information only of the selected type. You may create as many banks of registers as you like (within available memory).

Registers must be allocated and named before they may be used elsewhere in your Node.

If you are using ISaGRAF, you may use either the ISaGRAF Dictionary Tool or the ICL ScadaBuilder Register Allocation Tool. All ISaGRAF variables must have a network address (aka register address) to be used within ScadaBuilder. The address must be between 1 and 65535 decimal. In the case of Message Register, only network addresses of 1 to 9999 are allowed. No hex (base 16) numbers are allowed to be entered.



Note: A word of caution -- be careful re-numbering register addresses from the ISaGRAF tools. ScadaBuilder must reconcile any register that has moved from its previous position and may need to prompt the user when both the name and the register address are changed at the same time. Using the Register Allocation Tool within ScadaBuilder eliminates this problem.

Entering registers is easy in the ScadaBuilder Register Allocation Dialog. The only required parameters are the Prefix, and Index. Pressing Enter in any field is the same as clicking the Add button. The Index (aka network address) is automatically incremented at this time making the Prefix ready for the next register name. Just type it in and hit enter again to go to the next Index.

Entering large blocks of data can also be accomplished easily. Using any of the Enum fields allows the interaction of the Count and the enumeration of the register. For example, if the following data is entered:

Prefix:	Enum:	Template file:	Enum:	Suffix:	Enum:
pump	1	(none)		Run	
Sample output:		Index:	Count:		
pump1Run		1	4	<input type="checkbox"/> Retained <input type="button" value="Add"/>	

This will yield the following output:

pump1Run	1
pump2Run	2
pump3Run	3
pump4Run	4

Any Enum field may be used for the Prefix, Middle, or Suffix of the register name.

Type

Specifies the type of the register bank:

- Bool - 1-bit Boolean holds ON/OFF, 1/0 or TRUE/FALSE values.
- Int8 - 8-bit signed or unsigned integer (range -128 to 127 or 0 to 255).
- Int16 - 16-bit signed or unsigned integer (range -32,768 to 32,767 or 0 to 65,535).
- Int32 - 32-bit signed or unsigned integer (range -134,217,728 to 134,217,727 or 0 to 4,294,967,296).
- Float - 32-bit signed or unsigned single precision IEEE floating-point (range +/-3.4 x 10³⁸).
- Double - 64-bit signed or unsigned double precision IEEE floating-point (range +/-1.7 x 10³⁰⁸).
- Buffer - Array of bytes or characters.

Once a register bank has been created, its type cannot be changed arbitrarily -- only between compatible types:

- int8/int16/int32
- float/double



In ISaGRAF, only Booleans, Integers(32-bit), Reals(32-bit) and Messages(Buffers) are supported.

Block Size

Determines the number of registers allocated in the bank.

This parameter is not available in ISaGRAF

Prefix

Prefix is the first lettering to go into the register name when the add button is selected. In this example, the word "pump" will be prefixed into each register created.

Prefix:	Enum:	Template file:	Enum:	Suffix:	Enum:
<input type="text" value="pump"/>	<input type="text" value="1"/>	<input type="text" value="(none)"/>	<input type="text" value=""/>	<input type="text" value="Run"/>	<input type="text" value=""/>
Sample output:		Index:	Count:		
<input type="text" value="pump1Run"/>		<input type="text" value="1"/>	<input type="text" value="4"/>	<input type="checkbox"/> Retained	<input type="button" value="Add"/>

Hitting the ENTER key in this field is the same as hitting the Add button.

Prefix Enum

The Prefix Enum is the first number to go into the register name when the add button is selected. For each register created, the Prefix Enum will enumerate from the value given register by register. In this example, the number after the word "pump" will be added to each register created i.e. "pump1...", "pump2...", "pump3...", and "pump4...".

Prefix:	Enum:	Template file:	Enum:	Suffix:	Enum:
<input type="text" value="pump"/>	<input type="text" value="1"/>	<input type="text" value="(none)"/>	<input type="text" value=""/>	<input type="text" value="Run"/>	<input type="text" value=""/>
Sample output:		Index:	Count:		
<input type="text" value="pump1Run"/>		<input type="text" value="1"/>	<input type="text" value="4"/>	<input type="checkbox"/> Retained	<input type="button" value="Add"/>

Template File

Template files allow users to create and reuse a set of register names and configuration over and over again.

Template files are located in the <Install Directory>\Templates where the ScadaBuilder program has been installed. Each template file has an extension that denotes its type.

For example:

Real Bank	*.reals
Integer Bank	*.integers
Boolean Bank	*.booleans
Message Bank	*.messages

These files are stored in a .txt format and may be created and edited in Notepad.

The format for the file is

<RegisterName><R(Optional Retained)><"Comment Text"(Optional Text)>

where multiple lines mean multiple registers in the file. Here is an example network statistics file

TransmitCommand	, "Transmit Command"
ReceiveResponse	, "Receive Response"
ReceiveCommand	, "Receive Response"
TransmitResponse	, "Transmit Response"
ReceiveRoute	, "Receive Route"
TransmitRoute	, "Transmit Route"
ReceiveTimeout	, "Receive Timeout"
ChecksumError	, "Checksum / CRC Error"
BadContentError	, "Bad Content Error"
ConfigurationError	, "Configuration Error"
LostConnection	, "Lost Connection"
SuccessPercent	, "Success Percent"
LastReceivedCommand	, "Last Received Command"
CurrentPathID	, "Current Path ID"

The format is:

- One line per register.
- First contiguous set of letters is the name of the register (32 characters max)
- No special characters in the register name except an underscore
- An R anywhere on the line by itself or delimited will make the corresponding register retained
- Comments are in quotes
- Tab, Comma, Semicolon, or Space delimited

Middle Enum

When configured, adds an enumerated value after the <template> name of each register created. The suffix name and number may also be added if desired.

Suffix

This field appends text to the end of each of the register names created.

Suffix Enum

When configured, adds an enumerated value at the end of each register created.

Sample Output

The sample output show the result of user input from the Prefix, Enum, Template File, Enum, Suffix, Enum interface. User can see what the result of their input might look like before they confirm the addition of a block of registers.

In the case of a template file being used, the sample output will use only the word <template> for the register names from the template file.

Map to NVRAM (Make Retained)

If this option is checked, the register bank will be stored to NVRAM (Non-Volatile RAM). NVRAM retains its contents when power is removed from the Node.

This checkbox is not available when using ISaGRAF.

Register Name

The name to associate with the register. Registers are referenced by name throughout your ScadaBuilder Node.

In ISaGRAF, you have up to 32 alphanumeric characters to express your variable names.

It is suggested that as many characters as possible be used to describe the register's use. This will help in commenting and trouble shooting code.

Index

The index to associate with the register. Register indexes determine how registers are accessed via communications protocols (such as Modbus and BrickNet).

Indexes are the network addresses. In ISaGRAF, each index must be unique across all data types. For example, if you create an Integer register with an index of 1000, you CANNOT also have a Boolean or Real register with this same index. If you are creating a block of registers, enter the index number of the first register of the block into this field.

Indexes are always decimal numbers. They may range from 0 to 65535. Since the index is used by communications protocols for access to the data represented by this tag, it must be compatible with the protocols used. Note that the front header of the address specific to a protocol is omitted (i.e. 4001 instead of 34001 for Modbus).

Do NOT use an address of '0' for a register that is to be accessed via a communications port, even if it is valid for a specific protocol. An index value of 0 may only be used by ISaGRAF internal variables that will NOT be accessed by communications protocols or by ScadaBuilder functions.

ISaGRAF variables that are internal will show as a red "0" in the Registers window. This is okay so long as they are only used within ISaGRAF.

Count

The count allows a block of registers to be named at once when the 'Add' button is clicked. The specified register index will be automatically incremented for each register in the block.

A numeric value will be appended to the end of each register name (or from any Enum field) and is also automatically incremented for each register in the block. If the specified register name ends with a number, then it will be used as the starting numeric value. If the specified register name does not end with a number, then "1" will be used as the starting numeric value.

Messages Size

Number of characters in a Message buffer.

Retained Check Box

The retained box is used to signify those registers that are kept from runtime to runtime for the specific and configurable operation of a system. These registers will retain their value through a power cycle. During development, when registers may be added or deleted from the database, initial values may be used.

It is recommended practice to set initial values to a "known good" if not optimal state for the operation of the controller program. Should the retained values not "hold" through a program download for example, then the initial setpoint values will still allow the program to run.

The Retained box should be checked for the following data types:

Setpoints	Any register that operates on a level where the end user may change it for operational purposes
Timer Setpoints	Debounce, Delay timer setpoints and nuisance alarm time setpoints.
Initialization Booleans Or Setpoints	Registers that decide whether a controller has been initialized or not.
Phone Numbers	Any integer or message buffer that holds a phone number of any kind.
Runtime Counters	Pump or motor runtime accumulators
Start Accumulators	Pump or motor start accumulators
Flow Totalizers	Total gallons or other volumetric totalizers

Comments

Comments are descriptive text to help document the purpose or function of a register. The size of the comment text is limited to 60 characters for each register.

Add Button

The Add button adds a name/index pair(s) to the list.

File Button

The File button is used to import register names from a text file. Each line in the text file is considered one entry except for blank lines which are ignored. Each entry will be appended to the register name if one is specified. If the 'Count' value is non-zero, then the 'Count' determines how many registers are named. If the 'Count' value is zero, then the number of valid lines in the text file will determine how many registers are named.

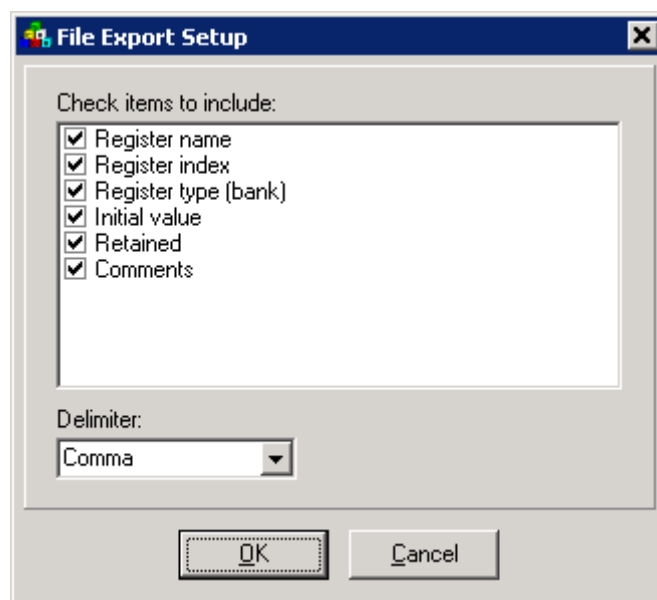
Register List Button

The Register List gives the user a complete list of all register names and indexes used in a separate window. This window may be left open while entering new registers to reference available register index and facilitate any indexing function. Some examples might be reference indexes for Modbus reads and writes or access the ISaGRAF dictionary by various read and write function blocks.

File Export Button

The File Export Button will allow a user to generate a delimited list of all registers in the system. This can be useful for importing data into a spreadsheet or SCADA system's tag database.

Clicking the File Export button will bring up the following dialog:



From here, the user can chose what fields to include in the file. The output is in text format.

Name List

This is the list of register name/index pairs that have been created and their associated attributes. The "I/O," "TUI," "CS", "NE" and "RS" columns indicate where a register can be used:

- I/O - Input/Output channels
- TUI- Textual User Interface
- CS - Communications Statistics
- NE - Network Events
- RS - Remote Scaling

Associating registers with specific uses helps to keep your project organized and keep the register lists shorter elsewhere in the program for easier selection.

I/O Checkbox

Register List Management

This checkbox indicates that the register may be mapped to an I/O board channel. If unchecked, the register name will not show up in the listbox when mapping an I/O channel.

TUI Checkbox

Register List Management

This checkbox indicates that the register may be used on a TUI (Textual User Interface). If unchecked, the register name will not show up in the TUI display controls where registers are used.

Comm Stats Checkbox

Register List Management

This checkbox indicates that the register may be used to store Network Session communications statistics/status values. When unchecked, the register name will not show up in any communications statistics register assignment lists.

Network Events Checkbox

Register List Management

This checkbox indicates that the register may be used in Network Events. When unchecked, the register name will not show up in any Network Event register lists for source or destination.

Remote Scale Checkbox

Register List Management

This checkbox indicates that the register may be used for Remote Scaling (in Network Sessions). When unchecked, the register name will not show up in the Remote Scaling record's register list.

Retained Register Drop Down List

For Pinnacle and later controllers there are two types of retained variables: Fast and File.

Fast Retained Memory:


Retained Registers that are stored in Fast memory get written to a small FRAM (Ferrous Random Access Memory) located on the CPU board. This chip can store up to 7K and should be used for:

Runtime Counters	Pump or motor runtime accumulators
Start Accumulators	Pump or motor start accumulators
Flow Totalizers	Total gallons or other volumetric totalizers

File Retained Memory:

Retained registers that are stored in File memory are written to the IDE flash disk whenever there is a value change. The interface is significantly slower than Fast retained memory but has the advantage of practically unlimited storage and retained files can be copied from one controller to another with the application files.

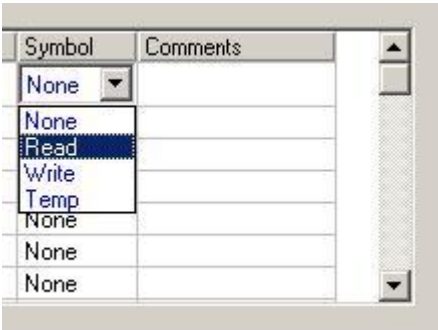
Setpoints	Any register that operates on a level where the end user may change it for operational purposes
Timer Setpoints	Debounce, Delay timer setpoints and nuisance alarm time setpoints.
Initialization Booleans Or Setpoints	Registers that decide whether a controller has been initialized or not.
Email Addresses	Used to send emails to particular recipients.
Phone Numbers	Any integer or message buffer that holds a phone number of any kind.



It is recommended that all retained values that affect the program at runtime such as those listed above be given an initial value that will get written to retained memory when the program is first started. These values should allow the program to run at startup as intended to prevent controller replacement from being more difficult than it has to be.

In the Pinnacle and later controllers, it is extremely difficult to remove a retained value once in the system. To reset the retained values back to initial values, ScadaBuilder provides a utility to do this. Go to the **Target | Reset Retained...** menu option. This will require a restart of the application or a power cycle to take effect.

Symbol Column



Symbols are the way that the User Portal web interface allows access to register if a user has Registers Permissions. To expose a register to the User Portal Live View, simply select one of the following options.

Read - Gives user read-only permissions for the register.

Write - Gives the user read/write access to the register.

Temp - Gives the user read/write access and the ability to abort a modification should a mistake be made. The user will also be able to confirm any change once the change has been tested.

For more information on setup, see *User Portal (Web Interface)* (on page 423) for details.

Format Button

The "Format" button opens the Format Editor window. The Format Editor allows you to define information that will be formatted and automatically written to a Message (buffer). The information can include static text as well as register values. Formatting only applies to buffer (message) variables.

Close Button

The close button simply closes this dialog. The data is saved as it is edited.

Show Attribute Columns

Show or hide the register list management checkboxes. Does not hide the retained checkbox.

The Attribute check boxes enable the user to tailor the register selections lists in various sections of ScadaBuilder. This simplifies “finding” the tag names that are needed in large ScadaBuilder configurations. The default is for all of the boxes to be checked (enabled), so you must uncheck them to remove them from appearing in a list.

I/O - When this box is checked, this register will show up in the selection lists under the I/O section.

TUI - When this box is checked, this register will show up in the selection lists under the Text User Interface (TUI) section.

CS - When this box is checked, this register will show up in the selection lists under the Communications Statistics section of Network Sessions.


NE - When this box is checked, this register will show up in the selection lists under the Network Events section of Network Sessions.

RS - When this box is checked, this register will show up in the selection lists under the Remote Scaling section of the I/O section.

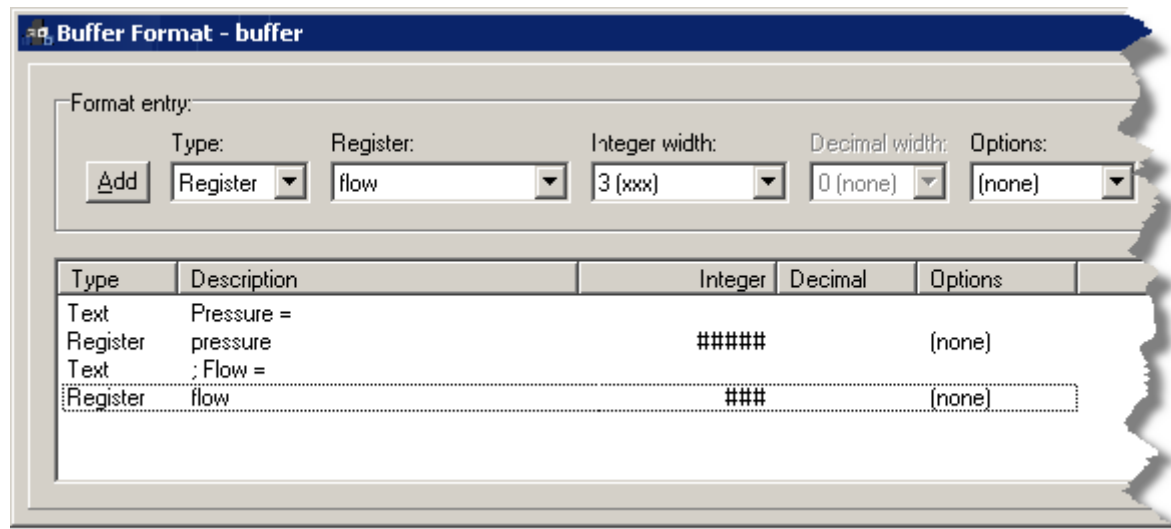
Buffer Format Editor

The Buffer Format Editor allows you to define information and associated formatting that will be written automatically to a buffer as the application runs. The information can include both static text and dynamic register values.

You can select items to be added to the format list. The resulting text string will be constructed from the items in the order listed and placed left to right into the associated message buffer.

To start, choose either "Text" or "Register" for the "Type" option. Additional fields will appear/disappear on the window depending on which is selected. Make additional choices as desired, then click the "Add" button to add the format item to the list. Repeat for as many format items as you wish. To find out more about a particular option, click the  button and then click the option.

Let's look at a specific example:



Type	Description	Integer	Decimal	Options
Text	Pressure =			
Register	pressure	#####		(none)
Text	; Flow =			
Register	flow	###		(none)

This will cause two register values to be written to the buffer, along with some label information. The resulting text that gets written to the buffer will look like this, if the pressure register value is "17" and the flow register value is "79":

Pressure = 17; Flow = 79

Note: once a format item has been put on the list, you can change its order or delete it by right clicking on the item.

Add Button

The "Add" button causes the format item to be added to the format list.

Type

Selects the type of format item to be added to the list.

Text A literal text string -- whatever you type in will be copied verbatim to the buffer.

Register A register value will be inserted into the buffer. The buffer will be updated as the application runs, so that changes in the register value will be reflected in the buffer.

Text

The literal text that will be inserted into the buffer.

Register

Selects the message register for which the value will be inserted into the buffer.

Integer Width

The number of digits of integer information to output to the buffer. Determines the number of digits shown to the left of the decimal point (if any) on floating point registers, or the total number of digits for an integer register.

Also determines the number of digits shown for a Boolean register. Boolean values are shown as "1" and "0."

Decimal Width

Determines the number of digits shown to the right of the decimal point for floating point registers.

Options

Selects additional options for generating the text associated with a register value.

Zero Pad If this option is selected, the value will be "padded" with leading zeros to the specified integer width.

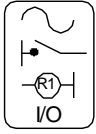
Blank if Zero If this option is selected, when the register value is 0, spaces will be output to the buffer.

Formatting Phone Numbers

"Zero pad" is useful for formatting a telephone number (last 4 digits) where the number might be 0041 but an integer register may only store "41". Adding the two zeros to "pad" the number would be done here.

"Blank if Zero" is particularly useful for area codes in phone numbers. If the area code is something other than 0 then it is added to the Message buffer; if not, then spaces are added and ignored when the sent to the modem.

Using I/O Channels and Mapping Registers



See *The ScadaBuilder Hierarchy* (on page 73)

All ICL controllers have internal inputs and outputs (I/O). Unlike a Programmable Logic Controller (PLC), the addresses of the I/O are not fixed; they can be “mapped” to any set of registers. In addition, most internal I/O is configurable. Analog inputs can be set to various modes and sensor types, and discrete inputs have hardware counters and signal conditioning that can be configured and/or mapped to registers. The I/O section of ScadaBuilder is used to scale, map and configure internal I/O. In addition, if remote I/O scaling is used, the scaling definitions for remote I/O are created and maintained in this section.

For the purposes of illustration, we will use an Etherlogic Ultima as it supports almost all I/O types.

To get to the I/O configuration screen, Expand the tree and open the I/O section in ScadaBuilder's Setup section:

Channel	AI	AO	DI	DO
1	AI1	AO1	DI1	DO1
2	AI2	AO2	DI2	DO2
3	AI3	AO3	DI3	DO3
4	AI4	AO4	DI4	DO4

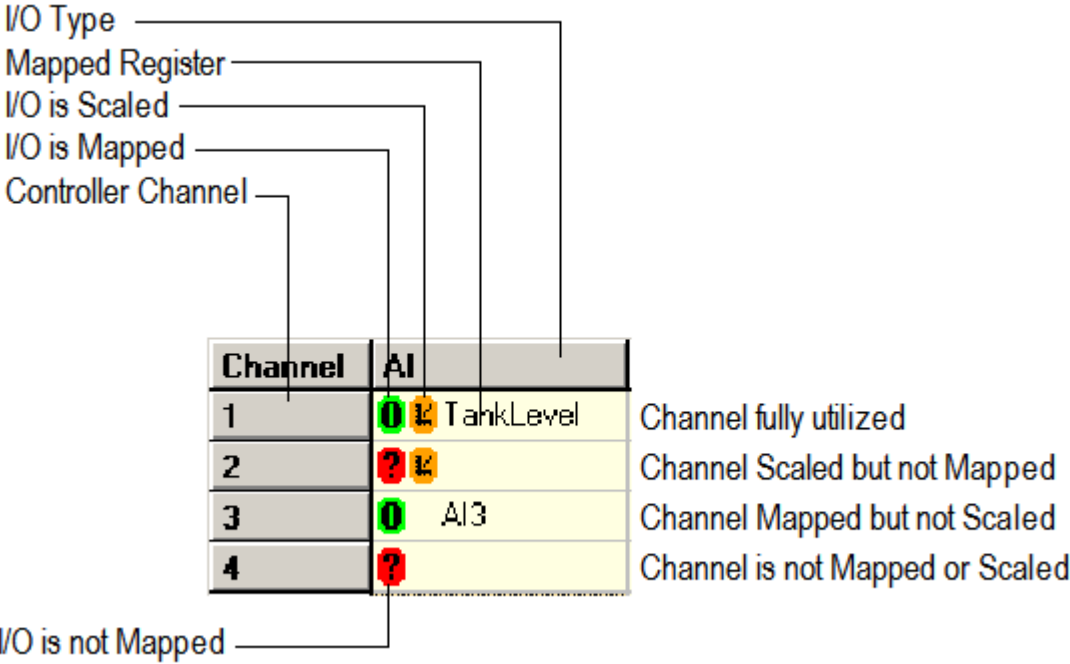
Map = AI1, Scale = 0 to 20mA

I/O options:

Watch dog timer: 0 (x10 ms) DI gate time: 0 (sec) TC/mV averaging: 10.0 sec AI AC Filter Mode: 60 Hz

Double click on Configuration and you will get the window above.

I/O icons are meant to be immediately informative about how each I/O point is configured.



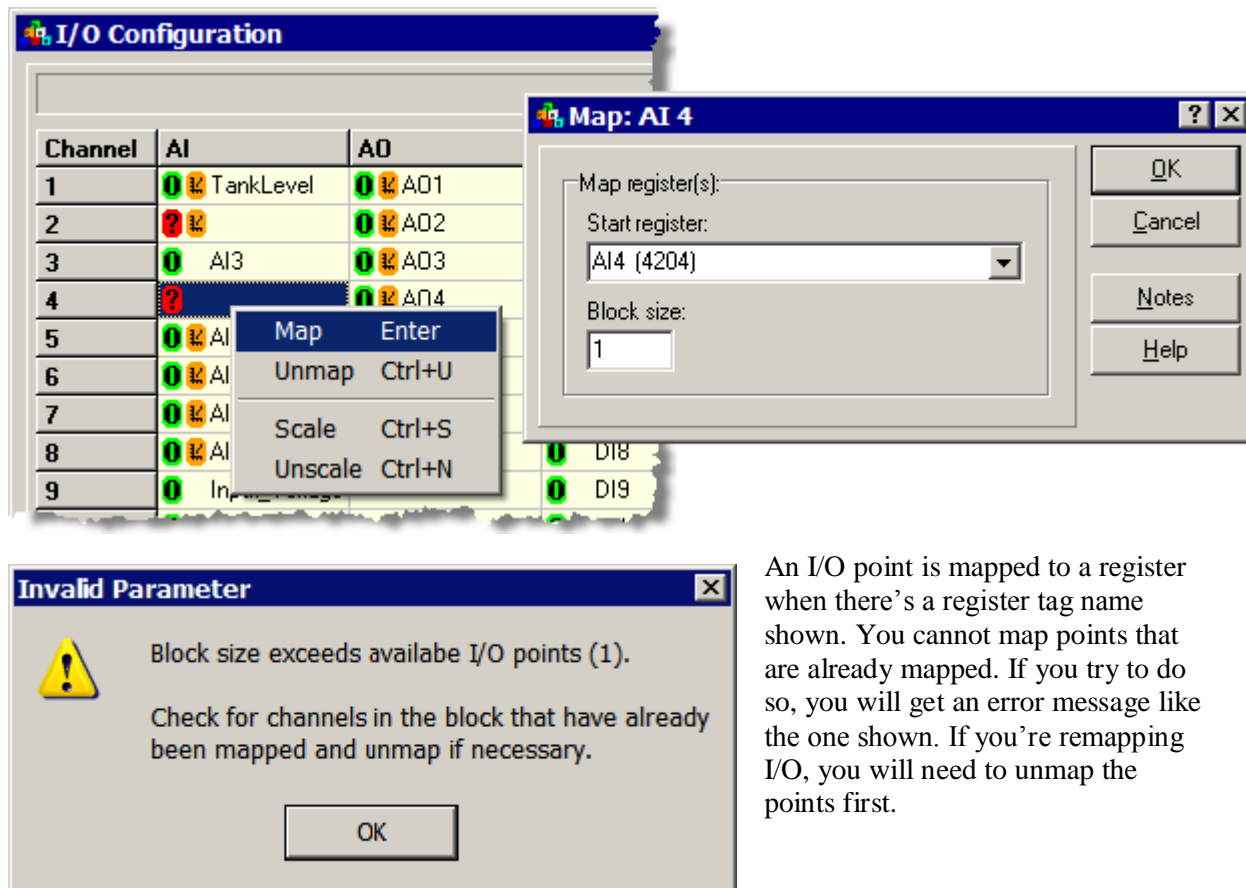
I/O Types	Description
AI	Analog Inputs represent real-world levels that can change in linear fashion.
AO	Analog Outputs for controlling real-world linear loads.
DI	Boolean Status Input. On or Off only.
DO	Boolean control point (Coil Output) for turning on relays or other boolean controls.
DI Count	Counters or Accumulators counting the number of positive transitions (false to true) of a DI.
DI Rate	Allows the count of positive transitions within the DI Gate time. If DI Gate Time is set to 1 second then DI Rate will be in Hertz.
DO Flash	While a DO is set to On, setting this channel will make the respective DO Flash (SFP only).
PI Count	Same as transitions above except it works with the Mag pickups.
PI Rate	Pulse input rate works with the Mag pickup changes and times from the DI Gate time.
Configurations	Description
AI Filter	In some cases, this will be a millisecond parameter. Others will simply require an On or Off state.
DI Filter	For channels that support high speed counting DI's, turning this on will slow the input back to standard frequency response (usually 40Hz or so).

In This Section

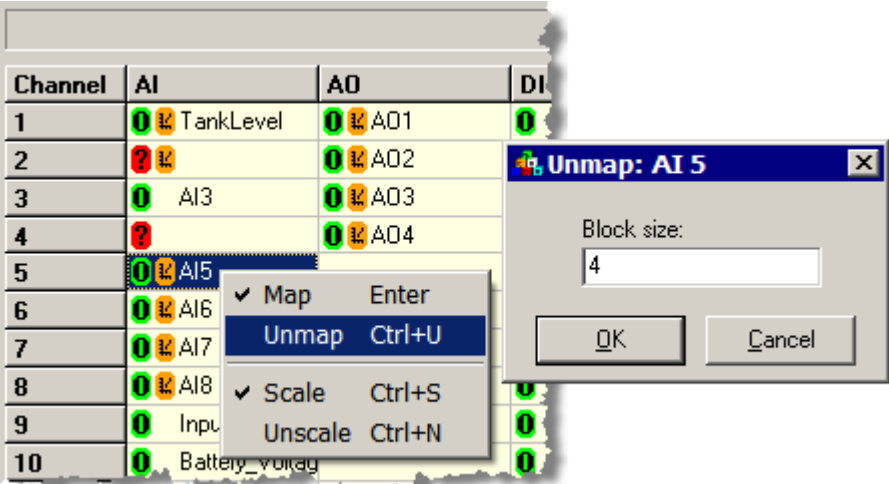
Mapping and Unmapping I/O to Registers.....	131
Applying a Scaling Record	135
I/O Options	136
I/O Configuration - ICL-4300 (PC-in-a-Brick) Controllers	137
I/O Scaling.....	138
I/O Ranges for Different Scaling Modes.....	141

Mapping and Unmapping I/O to Registers

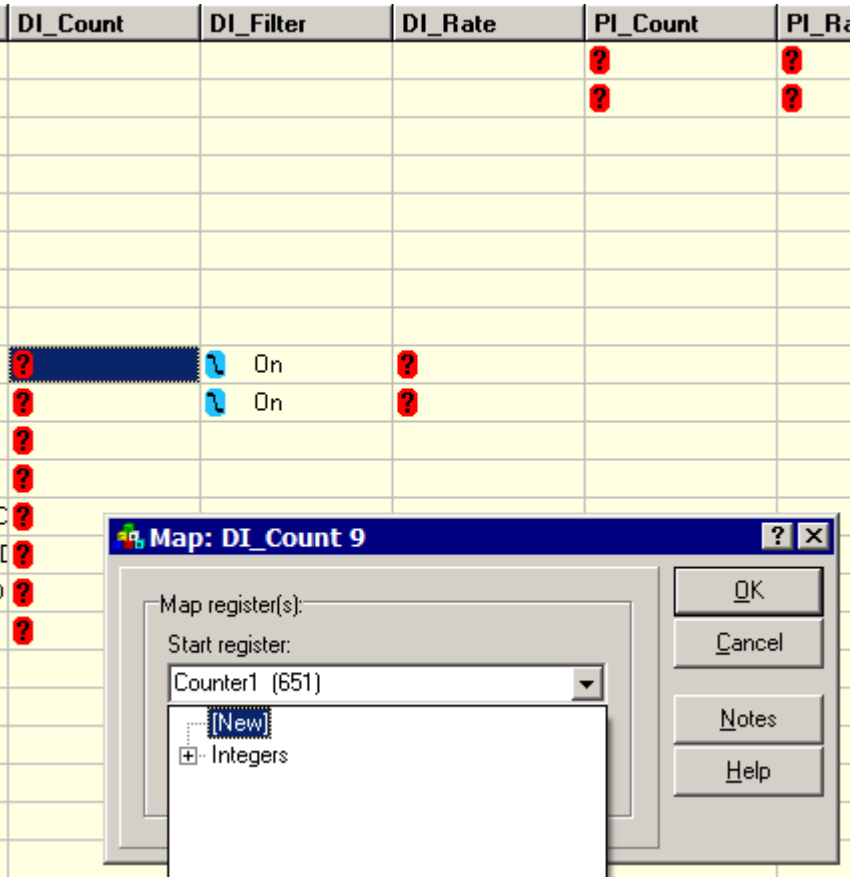
I/O can be mapped or unmapped to registers in blocks. To map one or more points to registers, simply double-click on the first point or highlight the first point and click on the “Map” button. Select the tag name of the first register that the block of I/O points is to be mapped to, enter the block size, and click on “OK”.



An I/O point is mapped to a register when there's a register tag name shown. You cannot map points that are already mapped. If you try to do so, you will get an error message like the one shown. If you're remapping I/O, you will need to unmap the points first.



To unmap I/O points, highlight the first point and click on the “Unmap” button. Enter the block size, and click “OK”.



Some controllers support hardware counters. These counters can be mapped or unmapped to registers in blocks just like I/O.

You can also create Registers on the fly from here is well by select [New] from the Mapping dialog to open the Registers configuration dialog.

EtherLogic Ultima									
Ch	AI	AO	DI	DO	DI_Count	DI_Filter	DI_Rate	PI_Count	PI_Rate
1	AI1	AO1	DI1	DO1					
2	AI2	AO2	DI2	DO2					
3	AI3	AO3	DI3	DO3					
4	AI4	AO4	DI4	DO4					
5	AI5		DI5	DO5					
6	AI6		DI6	DO6					
7	AI7		DI7	DO7					
8	AI8		DI8	DO8					
9	Input_Voltage		DI9	DO9		On			
10	Battery_Voltage		DI10	DO10		On			
11	Instrument_Power		DI11	DO11					
12	Cold_Junct_Temp		DI12	DO12					
13			DI13	Sensor_Pwr_Dsbl					
14			DI14	Modem_Pwr_Dsbl					
15			DI15	Status_Led_Dsbl					
16			DI16						
17			DI17						
18			DI18						
19			DI19						
20			DI20						
21			DI21						
22			DI22						
23			DI23						
24			DI24						
25			DI25						
26			DI26						
27			DI27						
28			DI28						
29			Power_Fail						
30			Battery_Low						
31			Sensor_Overload						


I/O options:

Watch dog timer: 0 (x10 ms)DI gate time: 0 (sec)TC/mV averaging: 10.0 secAI AC Filter Mode: 60 Hz

Notice that the DI_Count column only has points on DI's 9 through 16. These are the channels that support at least 40Hz counters.

On DI 9 and DI 10, there are also DI filter configurations when off, support up to 5kHz.

Check your Technical Reference Manual for the differences between controllers as the "shared" I/O points differ in each Etherlogic model.



The I/O configuration for most models is done when you create the Node. This map will tell you a lot about what the controller can do if you get to know it.

I/O Map Reference

The I/O Mapping window provides a means of mapping registers to I/O channels.

When a register is mapped to an input channel, the input state/value will automatically be stored in the register when the Node setup is running. When a register is mapped to an output channel, the value in the register will be automatically reflected on the output.

I/O Configuration - Map Button

Maps the selected I/O point to a register or a block of registers of the appropriate type.

I/O Configuration - Unmap Button

Unmaps the selected I/O channel or block of I/O channel from their respective registers.

I/O Configuration - Scale Button

Allows user to specify the I/O Scale record for any analog input or output. Can also operate on blocks of registers/channels.

I/O Configuration - Unscale Button

Allows user to unscale a single or block of analog input or output channels. Unscaled analog channels run in "Raw" mode.

Start Register

The starting register to map to the selected I/O channel.

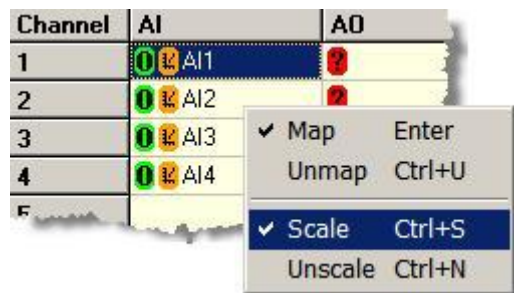
Block Size

The number of registers to map starting at the selected I/O channel.

Unmap Button

The "Unmap" button causes the register to be unmapped from the I/O channel. You can also specify a block size to unmap to do more than one point at a time.

Applying a Scaling Record



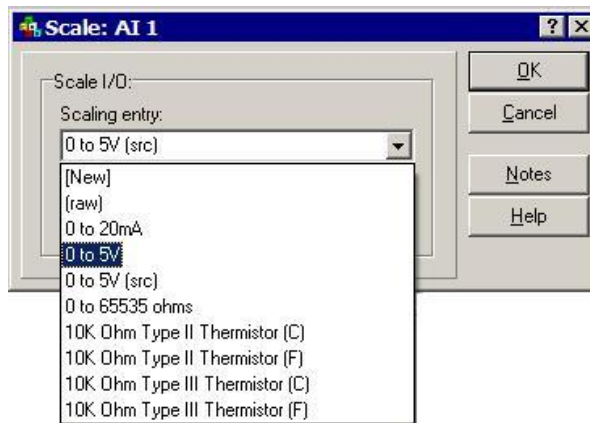
Select the I/O Channel you wish to scale.

Right click or click the Scale button on the right hand side.

If you already have a scale on your point, then you may need to unscale it first.

Select the scaling you want or create a [New] one. If you have a number of empty channels (not mapped), you may want to select a block size to save time.

Click OK.



I/O Scale Assignment Reference

The I/O Scale assignment window lets you assign a scaling to an I/O channel. This allows I/O values to be represented in "real world" units.

Scaling Entry

This is used to select the scaling entry that will be applied to the channel.

Block Size

The Block Size lets you apply the scaling to multiple channels.

Unscale Button

The Unscale Button removes the scaling from an I/O channel.

I/O Options

Of the I/O Options, some are shown at the bottom of the I/O mapping dialog others are not. Just like the I/O, Only those shown will apply to the Node Controller type configured.

I/O Configuration - Watch Dog Timer

The I/O watch dog timer specifies the maximum allowed duration for a communications transaction with the internal I/O processor. If the watch dog time expires then all of the discrete and analog outputs will be forced off until communications is restored.

I/O Configuration - DI Gate Time

The DI gate time specifies the time duration used in calculating digital input (DI) rates and pulse input (PI) rates.

I/O Configuration - DO Flash Rate

The DO flash rate specifies the on/off time interval when digital outputs are configured for flash mode.

I/O Configuration - Temperature Average Time

Specify the amount of time the I/O processor samples Temperature and millivolt mode sensors before updating the mapped registers. This parameter affects ALL sensors of the above two types.

I/O Configuration - AI AC Filter Mode

The A/D on some models has built-in AC filtering. When available, this configuration sets the filter to either 50 Hz or 60 Hz rejection depending on the installation location's AC power source.

I/O Configuration - ICL-4300 (PC-in-a-Brick) Controllers

The ICL-4300 is the one controller that ScadaBuilder cannot provide a “default” configuration for when a node is created since the I/O is made up of plug-in cards that the user selects. The user **MUST** therefore manually configure the I/O using the I/O Configuration window.

Because of the controller’s modularity, the I/O Configuration window for the ICL-4300 controller is unique with four I/O card selection fields at the bottom of the window, and four tabs to select the I/O configuration for each slot’s I/O card. Other than these differences, the mapping and scaling of the I/O is identical to the other ICL controllers.

Slot 1Slot 2Slot 3Slot 4

Slot1: Combo - 24

Channel	AI	AO	DI	DO	AI_Filter
1	0 AI1	0 AO1	0 DI1	0 DO1	Off
2	0 AI2	0 AO2	0 DI2	0 DO2	On
3	0 AI3		0 DI3	0 DO3	Off
4	0 AI4		0 DI4	0 DO4	On

I/O board types:

Slot 1:Slot 2:Slot 3:Slot 4:

Combo - 24DI 16 - 24[none][none]

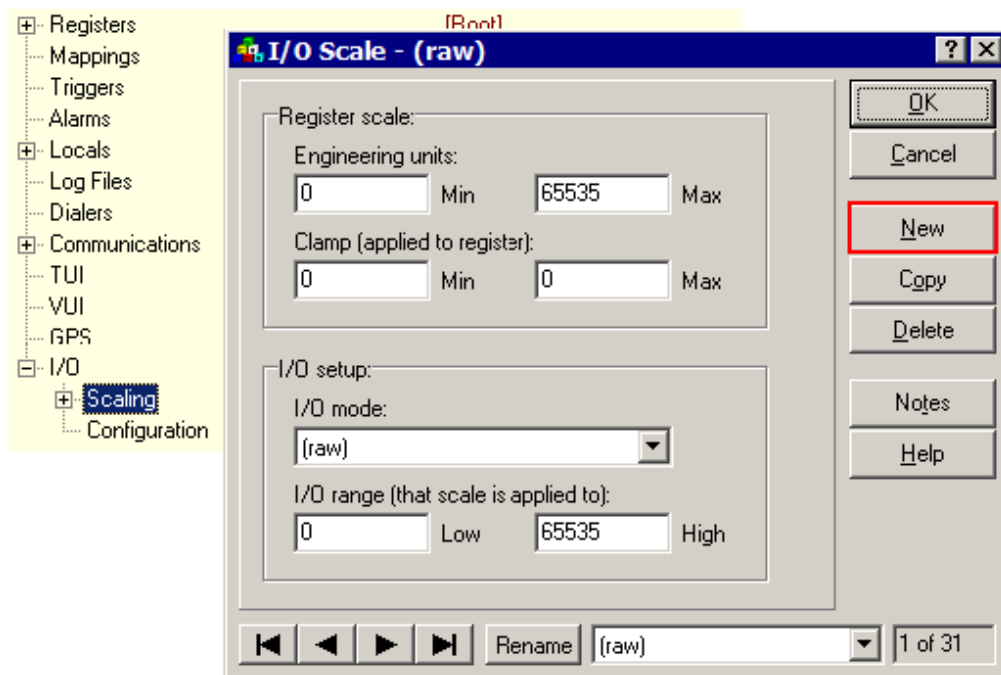
The I/O mapping features of each I/O card are summarized in the table below.

	[-----I/O Mapping -----]					[---- Configuration ----]	
	AI	AO	DI	DO	CNTR	AI Filters	DI Filters
DI16 (all)			X				
DO8 (all)				X			
DIO16-24			X	X	X		X
AI16	X					X	
AO16 (all)		X					
COMBO (all)	X	X	X	X			

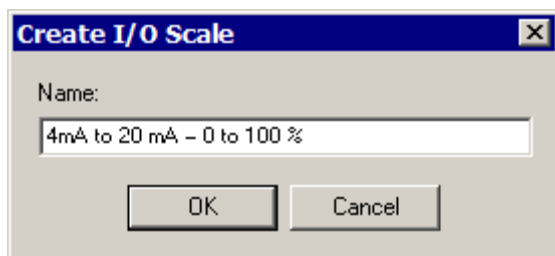
I/O Scaling

ScadaBuilder enables any analog I/O point, local or remote, to be scaled, eliminating programming of scaling math calculations. Frequently, the same scaling is applied to multiple I/O points, so ScadaBuilder uses scaling definitions that can be associated with one or more input or output points. There are some “pre-made” scaling definitions based on the specific controller hardware, but you can also easily define your own. The pre-made scaling definitions simply perform a one-to-one scaling unless modified. For example, in a ScadaFlex Plus controller, a 20mA analog input will read from 0 to 20000 from the I/O processor for a 0 to 20mA input signal. The pre-made definition scales this one-to-one so that without modification, the register values read by a controller program will be 0 to 20,000. For example though, you might want to limit the lower portion of the input range for a “normal” 4 to 20mA input signal and scale this value so that 4mA is 0 and 20mA is 100.00 (percent). Since the pre-made scaling is designed for 0 to 20mA, you will probably want to create your own 4 to 20mA scaling definition.

To create a new scaling definition, click on “I/O”, then double-click on “Scaling” in the Project Manger, or go to the **Setup | I/O Scaling** menu near the bottom of the menu.



Click on “NEW” in the scaling window. A dialogue window will pop up for naming the scaling definition. Accept the default name or enter a new one.



The more descriptive your name, the easier it will be to find it in a long list of names later on.

Once the scaling definition is named, a window like the one on the left is displayed. This example is for an EtherLogic Ultima controller.

When the analog input of an EtherLogic Ultima controller is configured for a 20mA input, the values that the I/O processor will report for an input of 0 to 20mA is 0 to 20000.

In this example, the input scaling compensates for the lower 4mA (value of 4000) and scales the input so that with a signal range of 4mA to 20mA, the controller register will see a reading of 0 to 100. Notice that the clamping has been set to a minimum of 0 and a maximum of 0, the default clamp values (the Engineering Min and Max) are used. If the input signal drops below 4mA, the reading will not go below 0. If the input reading goes above 20000 the register will stick (clamp) at 100.

To utilize a Scaling record, see *Applying a Scaling Record* (on page 135).

Register Min & Max

The Register Min and Max settings determine the scaled operating range. The Min corresponds with the low operating range limit of the analog input or output channel. The Max corresponds with the maximum signal level of the analog input or output.

Clamp Min & Max

The Clamp Min & Max parameters are used to clamp, or limit the resulting scaled values to the specified Min and Max values. To disable clamping, put 0 for both the Min and Max values.

I/O Mode

Selects the operational mode of the board with which the scaling entry will be associated. This will set the default values for the I/O range that corresponds to the selected mode.

The "(raw)" mode allows the register to be scaled with reference to the raw analog values (no calibration is applied to the analog converter values). This only applies to controllers that use internal serial I/O.

The "(none)" mode allows the register to be scaled with reference to the uncalibrated analog values (no calibration is applied to the analog converter values). This only applies to ICL-4300 model controllers.

The "(remote)" mode is used to scale registers that are accessed through communications (such as Modbus or Bricknet). This allows remote I/O to be scaled locally.

All other modes are controller specific. Different controllers will present different values to the scaling record. These values will fill in automatically when the I/O mode is selected.

For example, selecting 0-20 mA mode on an Ultima will fill in the values of 0 - 20000 in the I/O Range fields. Selecting the same mode on an Etherlogic LC will fill in values of 0 - 2000 showing the difference between the 16 bit A/D of the Ultima and the 10 bit A/D of the Etherlogic LC. Please see *I/O Ranges for Different Scaling Modes* (on page 141) for more information.

I/O Range Low & High

For the corresponding I/O mode, sets the end points that the Engineering Units Min and Max values correspond to.

Example 1:

You are setting up scaling for a 0 to 5V analog input card such that you want 0V to correspond to 32 degrees and 5V to correspond to 212 degrees. You would set the Engineering Units Min and Max to 32 and 212, respectively. Set the I/O Mode to 0 to 5V, and set the I/O Range Low and High settings to 0 and 5000 (or 50000), respectively.

Example 2:

You are setting up scaling for an analog output card such that 4mA should correspond to 0 liters per minute and 20mA should correspond to 100 liters per minute. You would set the Engineering Units Min and Max to 0 and 100, respectively. Set the I/O Mode to 0 to 20mA, and set the I/O Range Low and High settings to 400 (or 4000) and 2000 (or 20000), respectively.

Please see *I/O Ranges for Different Scaling Modes* (on page 141) for more information.

I/O Ranges for Different Scaling Modes

ICL product lines are not all the same when it comes to I/O. The different controllers support different ranges of Inputs (and outputs).

The following table lines out the supported sensor types and the I/O ranges that come back from the I/O system.

- **EL 10bit** are Etherlogic STD, Etherlogic Current Loop and Etherlogic LC.
- **EL 16bit** are Etherlogic Ultima, Etherlogic Integra, and Etherlogic Advanta.
- **SF+** is the ScadaFlex Plus Ethernet and ScadaFlex Plus OP/IO

I/O Mode	Minimum	Maximum	EL 10 Bit	EL 16bit	SF+
0 to 20mA	0	2000	X		
0 to 5Vdc	0	5000	X		
0 to 20mA	0	20000		X	X
0 to 5Vdc	0	50000		X	X
+/- 300mV	-30000	30000		X	
Resistance (ohms)	0	65,535	X	X	
Type J Deg C -240.7°C to 1199.0°C	-2407	11990		X	
Type J Deg F -401.2°F to 2190.2°F	-4012	21902		X	
Type K Deg C -261.2°C to 1369.5°C	-2612	13695		X	
Type K Deg F -438.1°F to 2497.1°F	-4381	24971		X	
Type T Deg C -263.2°C to 398.8°C	-2632	3988		X	
Type T Deg F -441.7°F to 749.8°F	-4417	7498		X	
Type E Deg C -267.4°C to 999.0°C	-2674	9990		X	
Type E Deg F -449.3°F to 1830.2°F	-4493	18302		X	
Type R Deg C -43.1°C to 1759.8°C	-431	17598		X	
Type R Deg F -45.5°F to 3199.6°F	-455	31996		X	
Type S Deg C -41.3°C to 1759.1°C	-413	17591		X	
Type S Deg F -42.3°F to 3198.3°F	-423	31983		X	
Type B Deg C 253.4°C to 1792.1°C	2534	17921		X	
Type B Deg F 488.1°F to 3257.8°F	4881	32578		X	
Type N Deg C -255.4°C to 1296.8°C	-2554	12968		X	
Type N Deg F -427.7°F to 2366.2°F	-4277	23662		X	
10K Thermistor, Type II Deg C	-401	2034	X	X	
10K Thermistor, Type II Deg F	-400	3981	X	X	
10K Thermistor, Type III Deg C	-401	2011	X	X	
10K Thermistor, Type III Deg F	-400	3939	X	X	
10 ohm .00427 RTD Deg C	-1900	2590		X	
10 ohm .00427 RTD Deg F	-3100	4820		X	
100 ohm .00385 RTD Deg C	-1900	2500		X	
100 ohm .00385 RTD Deg F	-3100	4820		X	
100 ohm .00392 RTD Deg C	-1989	8694		X	
1,000 ohm .00392 RTD Deg C	-1989	8694		X	

I

Using Triggers



See *The ScadaBuilder Hierarchy* (on page 73)

Triggers are pre-defined conditions that can activate ScadaBuilder functions. Triggers can generate alarms, and initiate dialed voice messages, pager alerts, text messages to cell phones, FTP transfers and e-mails. Triggers can also cause data to be written to log files and messages to be sent.

Triggers monitor the data in all types of registers.

For example, a single Trigger can be set up to look for a change of state in a block of several hundred points. This feature makes Triggers ideal for SCADA systems. By configuring only a few Triggers, an RTU can be created that sends a message on a change of state of the RTU's analog and discrete I/O points or efficiently logs data by only recording the data as changes occur.

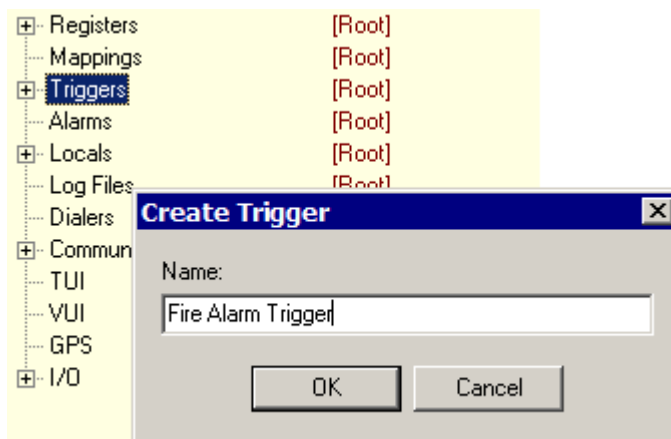


A Trigger can be configured to monitor a single point, or a block of points.

Triggers can be generated based on:

- sequential operations (cyclic Network Events only)
- time (periodic from internal millisecond timer or from Real Time Clock)
- time of day
- a change to ON, or OFF, or BOTH (state)
- a change in a numeric value (delta)
- a change in a buffer string
- a message updating register values (Network Session)
- changing values or activating a “button” on a local HMI screen (TUI write and TUI Button).

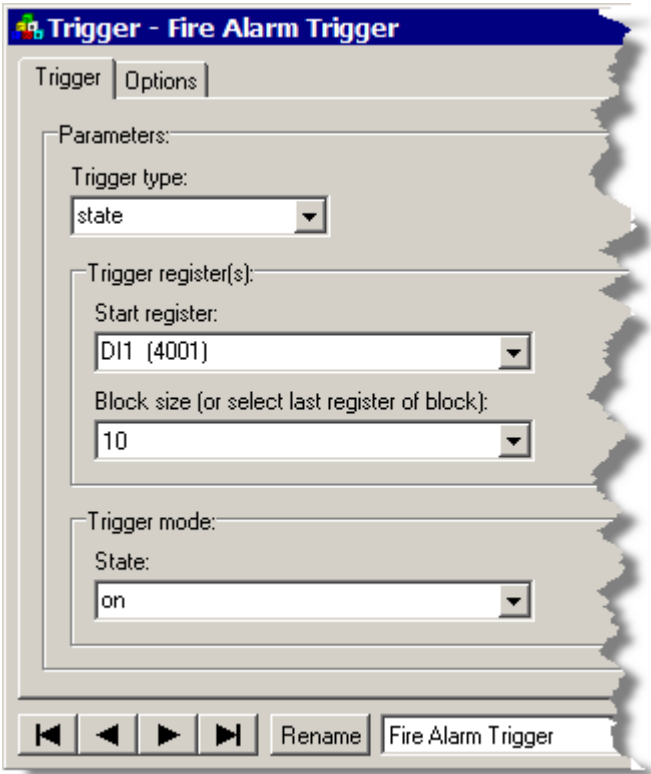
To create a Trigger, double-click on “Triggers” in the Project Manager (if there are no other Triggers) or click on “NEW” in the Triggers window. A window will pop up for naming the Trigger. Accept the default name or enter a new one.



In This Section

Defining a Trigger.....	144
Setpoint Triggers.....	145
Alarm Suitable Triggers	146
Special Trigger Types.....	147
Triggers Reference	148

Defining a Trigger



Available Triggers and configuration information:

Depending on the type of Trigger selected, the lower portion of the window changes to offer selections appropriate for that type of Trigger.

This example is for a State Trigger. The controller will monitor 10 discrete inputs, beginning with DI1 (the default tag name for Discrete Input #1). The trigger will be activated when any of the subsequent inputs turn ON. To create another Trigger, click on the “New” button and repeat the process.

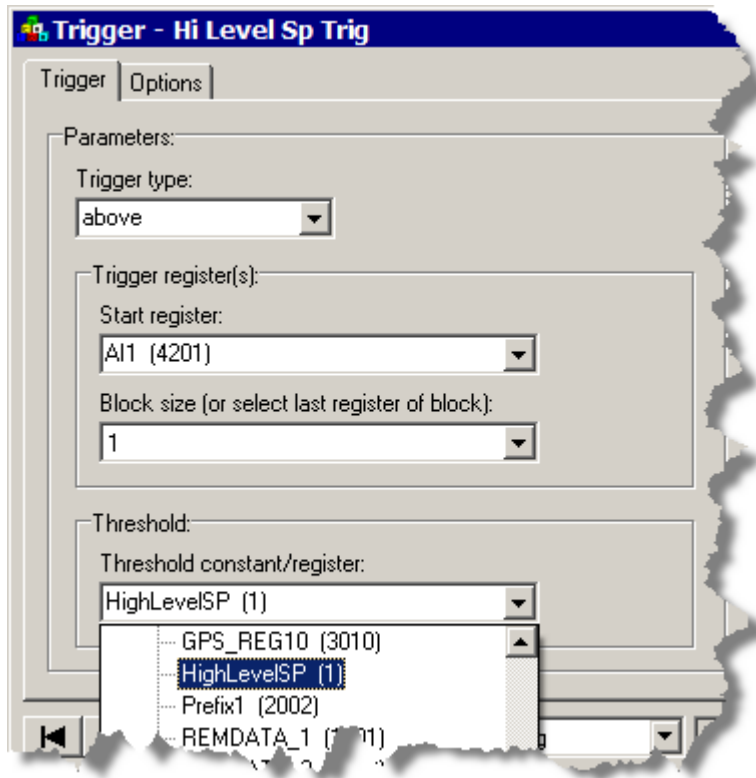
<i>Trigger</i>	<i>Data Types</i>	<i>Applications</i>	<i>Configuration</i>
CYCLIC	None	Network Events	number of cycles
TIMER	Time	All	seconds
STATE	Booleans*	Alarms and change detection	On, Off, Either, debounce on, debounce off, retrigger
DELTA	Integer's, Reals*	Change detection	Retrigger
ABOVE	Integer's, Reals*	Alarms and setpoint detection	Retrigger, Hysteresis
BELOW	Integer's, Reals*	Alarms and setpoint detection	Retrigger, Hysteresis
NET SESSION	All	Communications change detection (Scada)	Retrigger
BUFFER	Messages/Buffers	Datalogging and diagnostics	Retrigger
TUI WRITE	All	User change detection (Scada)	Retrigger
TUI BUTTON	User Interface	User Interface	None (use TUI editor)
RTC TIME	Real Time Clock only	Data Logging, Report Transmission	Period and duration.



NET SESSION Triggers are generated when at least one value in a designated block of registers is changed by an incoming communications message.

TUI WRITE Triggers are generated when at least one value in a designated block is modified from a TUI screen.

Setpoint Triggers



Three trigger types can be controlled via a setpoint register. They are

- *Delta*
- *Above*
- *Below*

Setpoints can be configured in the Trigger dialog by clicking on the Threshold field a selecting a register from the list. Only Integer and Real register will be available.



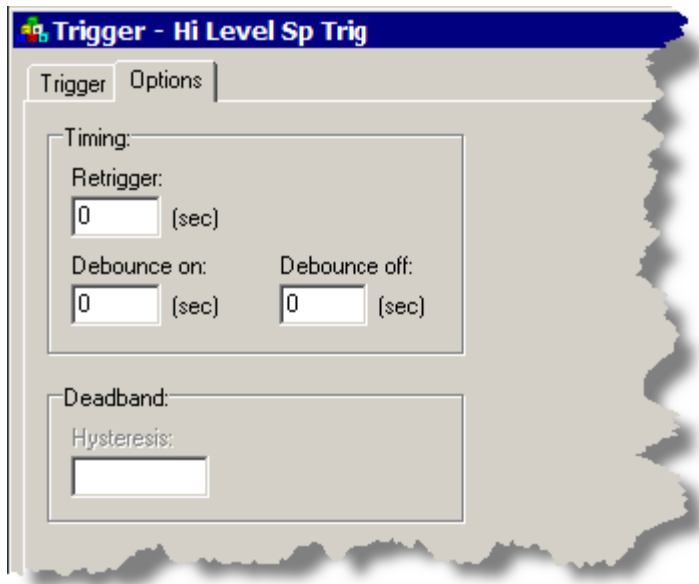
It is recommended that all registers used as a Threshold setpoint be set to retained or given an appropriate initial value or both.

Alarm Suitable Triggers

Alarming will be covered in the next section, but it is good to mention now that there are three Trigger types that should be used with Alarming in ScadaBuilder. Because Alarms have three states, Unacknowledged, Acknowledged, and Idle, the Trigger used to fire the Alarm must be of a type that retains a state that can be cleared. Triggers that possess this property are *State*, *Above* and *Below*

While the Trigger is in its configured "Triggered" state, the Alarm can only be acknowledged and will not go to idle. Once the Triggered state is removed, the Alarm may then return to the Idle state.

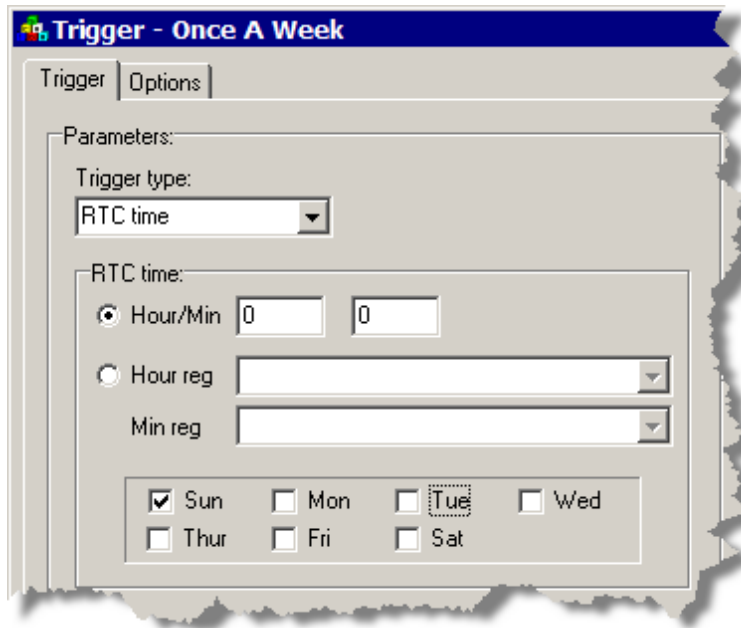
Trigger Options



ScadaBuilder Triggers that “look” at registers (DELTA, ABOVE & BELOW) have configurable “debounce” and “deadband” to avoid false triggers.

Also to prevent nuisance Triggers, the Retrigger parameter may be used to suppress the next Trigger for a period of time. Once a Trigger has been fired, the Trigger system will wait the Retrigger time before it will look at the Trigger condition again.

Special Trigger Types



There are special conditions in which the user may wish for something to happen.

For example, Triggering the transfer of a log file once a week (Sunday at midnight) can be done with a Real Time Clock Trigger:

See **Triggers Reference** (on page 148) section for other special Trigger types.

Another special Trigger type is the TUI Button. See **TUI Controls** (on page 399) for more information.

Triggers Reference

A Trigger specifies conditions which can be used to activate Network Events, Log Events, Log Archives, Alarms, and Local Events.

Type

Specifies the conditions for which the trigger is activated.

- | | |
|---------------|--|
| <i>Cyclic</i> | A Cyclic Trigger is activated whenever the Cycle Count is reached. A cycle is complete when every pending Network Event for the same device has been serviced. A Cycle Count value of 5 would cause the Trigger to be activated every 5 cycles. |
| <i>Timer</i> | A Timer Trigger is activated every time the specified number of seconds passes. The timer can either be based on the system tick or real-time clock. |
| <i>State</i> | A State Trigger can also be an edge trigger, the distinction being important only for alarms. A State Trigger is activated when the Boolean register (or block of registers) being monitored reaches the specified on/off state. For purposes of alarming, the condition is considered still active if the register is still in the specified state. The determining factor is whether the debounce parameter is set or not. If not, the Trigger acts like an edge trigger, if so, the Trigger waits the debounce time before firing. While the state condition is true, the associated alarm will remain and never go to an Idle state. |
| <i>Delta</i> | The Delta Trigger monitors a block of registers for a change equal to or greater than the specified threshold. The comparison values are captured each time the Trigger is activated. The threshold value can either be a constant or can be stored in a register. |

<i>Above</i>	The Above Trigger is activated when the register value (or block of registers) being monitored rises above the specified threshold. The threshold value can either be a constant or can be stored in a register. This type of trigger is often used to activate an Alarm. For purposes of alarming, the condition is considered still active if the register is above the specified threshold.
<i>Below</i>	Similar to the Above Trigger, except that the value (or block of registers) being monitored must fall below the threshold for the trigger to be activated. This type of trigger is often used to activate an Alarm. For purposes of alarming, the condition is considered still active if the register is above the specified threshold.
<i>Network Session</i>	The Network Session Trigger is activated when any register in the specified block is written to from a given Network Session. If an Address is specified, the Trigger activation is limited to writes that come from the specified address.
<i>TUI Write</i>	The TUI Write Trigger is activated when any register in the specified block is modified by a user via a TUI (Textual User Interface).
<i>RTC Time</i>	The RTC Time Trigger is activated when the real-time clock reaches the specified time. The trigger time (hours/minutes) can be specified as either a constant or in registers. This can also be utilized to start a periodic triggering based on the Real Time Clock (see Options Tab).
<i>Buffer</i>	A Buffer Trigger detects when a buffer is written to from an ICL configuration in Scadabuilder. Only one trigger per buffer may be configured.

Any trigger may be used in multiple Events of any kind.

Cycle Count

For Cyclic Triggers, this parameter determines how frequently the Trigger should be activated relative to other Cyclic Triggers. For instance, if Cycle Count is set to 1, the Trigger will be activated every cycle. A value of 6 will cause a Trigger to be activated 1/6th as often as those Triggers that are set to a Cycle Count of 1.

The "Cycle" only applies to Network Event List cycle.

Timer Period

This parameter determines how frequently a Timer Trigger should be activated. This can operate from the internal millisecond timer or the Real Time Clock.

Timer Mode

This timer mode determines the resource used for activating the Timer Trigger. The options are as follows

System Tick

The system tick is generated from the controller's processor clock. This may have some drift as compared to the real time clock, but is a more efficient resource to use.

Real-Time Clock

The real-time clock is a peripheral timing device. It is more suited for triggers that are used for data logging and time stamps, but requires more system resources to operate.

Start Register

For change-based Triggers, defines the starting register of the block of registers to monitor for change. This applies to Triggers of the following types:

- State
- Delta
- Above
- Below
- Network Session
- TUI Write

Register Block Size

For change-based Triggers, defines the size of the block of registers to monitor for change. This applies to Triggers of the following types:

- State
- Delta
- Above
- Below
- Network Session
- TUI Write

State

Specifies the value of the register that activates a State Trigger.

On

The on (or true/non-zero) state activates the Trigger.

Off

The off (or false/zero) state activates the Trigger.

State triggers are able to utilize a debounce time for both on and off states to prevent nuisance triggers. A value of zero in the debounce field will act as though the state triggers works off of the positive (On) or negative going (Off) edge.

Threshold Constant

For change-based Triggers, sets the threshold that the register value is compared against. This applies to Triggers of the following types:

- Delta
- Above
- Below

Threshold Register

For change-based Triggers, specifies the register which holds the threshold that the register value (usually retained) is compared against. This applies to Triggers of the following types:

- Delta
- Above

- Below

Buffer

Buffer from which to detect a change.

Network Session

When register write messages to the specified register block are received on this Network Session, the Trigger will be activated.

Network Address

Qualifies a Network Session Trigger so that only register writes from a network node with this address will cause the Trigger to be activated.

Real Time Clock (RTC)

This allows the user to trigger on a particular hour and minute of a particular day of the week or everyday if they so choose.

The hour and minute triggering time may also be mapped to and set from registers.

Hour/Min Constants

For RTC Time Triggers, sets the hours and minutes of the time to activate the trigger.

Hour/Minute Registers

For RTC Time Triggers, specifies the registers (usually retained) that hold the hours and minutes values of the time to activate the trigger.

Day Checkboxes

For RTC Time Triggers, these checkboxes select which days-of-the-week the trigger should be activated when the specified RTC time is reached.

Trigger - Options Tab

Trigger Options are applied depending on Trigger Type.

Trigger Enable Control Map

Allows you to map a boolean register to enable and disable the trigger dynamically from a control program. TRUE = enabled, FALSE = disabled.

Retrigger Time

The Trigger will not be activated more frequently than the Retrigger Time. The Retrigger Time may be used to eliminate nuisance alarms and excessive change-based communications.

During the Retrigger time, the Trigger is NOT evaluated so any offending conditions will be ignored..

Debounce Time

The Trigger condition must be present for the Debounce Time before the Trigger is activated.

The Debounce Time may be used to eliminate nuisance alarms and excessive change-based communications.

Debounce may be defined for ON time or OFF time or both.

Deadband Hysteresis

Used with Above and Below Triggers. The Hysteresis parameter helps avoid nuisance Trigger activations. Before the Trigger can be reactivated, the register value must cross the Trigger Threshold and return beyond the Hysteresis point, then cross the Threshold into the active range again.

For example: an Above Trigger is defined, with the Threshold set to 10 and the Hysteresis set to 2. When the register value rises above 10, the Trigger will be activated. Before the Trigger can be reactivated, the value must drop below 8 ($10 - 2$), then rise above 10 again.

RTC Retrigger Options

While the Hours and Minutes parameters of an Real Time Clock (RTC) Trigger control at what time of day or day of week an event happens, the Retrigger Increment and Retrigger Duration can expand that functionality to trigger an event periodically after the Hours and Minutes times has been reached. For example, you could configure a trigger to start at midnight and retrigger every 15 minutes after that for a period of time—say two hours.

In short The Retrigger Increment tell how often to trigger in hours and minutes after the RTC event occurs while the Retrigger Duration defines how long in hours and minutes to continue that retriggering.

In another example, a user may wish to have a new log entry triggered every half an hour on the half hour. Doing this is easy. Set up an RTC trigger on the next hour; say 0 hours and 00 minutes (midnight). Then set the Retrigger Increment hours to 0 and the Retrigger Increment Minutes to 30. Set the Retrigger Duration Hours to 23 and the Retrigger Duration Minutes to 59 to make sure that the duration covers the rest of the 24 hour period.

Using Local Events

ocal Events and AlarmsFILE3593

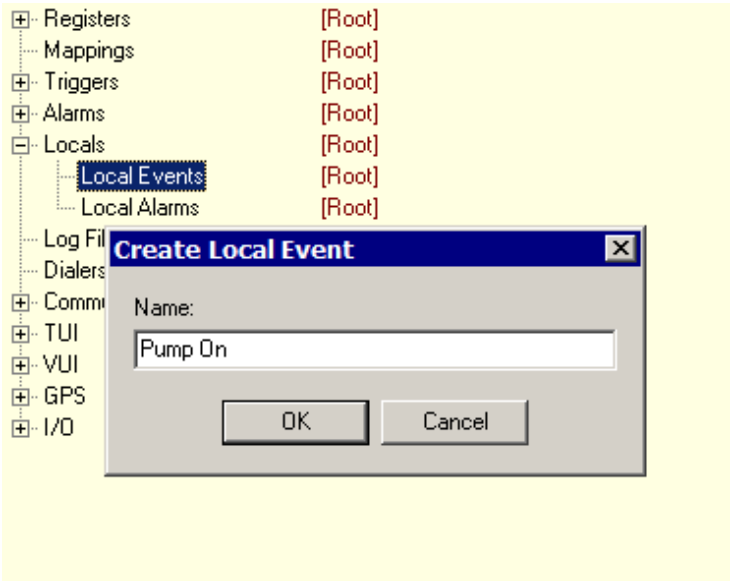
In This Section

Creating A Local Event	155
Defining A Local Event.....	155
Local Events Reference	156

Creating A Local Event

To create a Local Event, double-click on “Locals”, and then “Local Events” in the Project Manager (if there are no other Local Events defined) or click on “NEW” in the Local Events window.

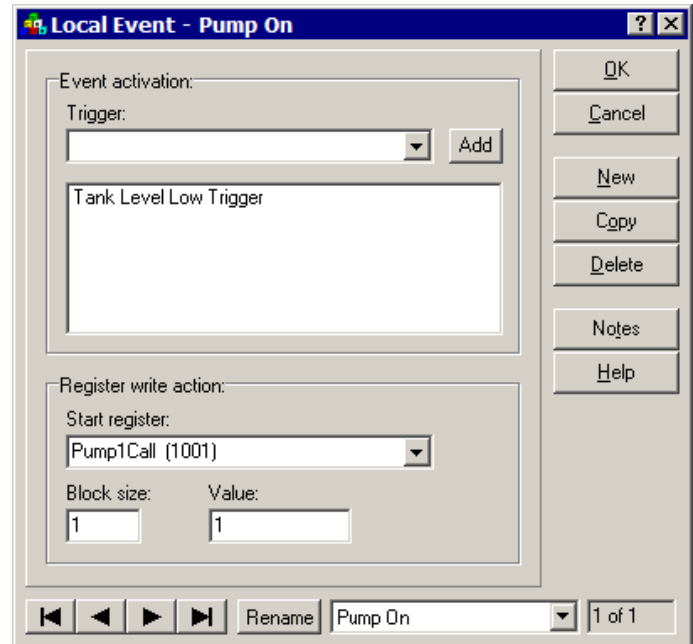
A dialogue window will pop up for naming the event. Accept the default name or enter a new one.



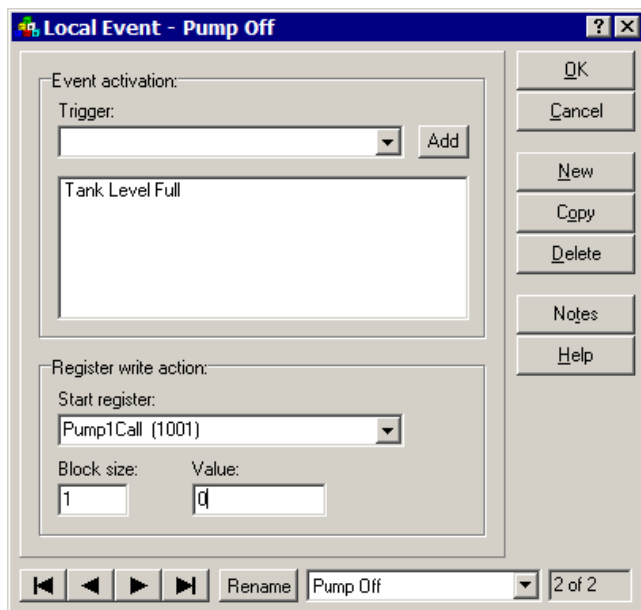
Defining A Local Event

Once the event is named, a window for defining the Local Event is displayed.

A Local Event requires one or more Triggers. In this case, a Trigger from an analog sensor that goes below a setpoint will turn on a pump connected to Pump1Call which is in turn mapped to a Digital Output.



The 'Local Event - Pump On' dialog box is shown. It has a title bar with a question mark and close button. The 'Event activation:' section contains a 'Trigger:' dropdown menu with an 'Add' button. Below it is a list box containing 'Tank Level Low Trigger'. The 'Register write action:' section contains a 'Start register:' dropdown menu with 'Pump1Call (1001)' selected. Below this are 'Block size:' and 'Value:' fields, both containing the number '1'. On the right side, there are buttons for 'OK', 'Cancel', 'New', 'Copy', 'Delete', 'Notes', and 'Help'. At the bottom, there are navigation arrows, a 'Rename' button, a dropdown menu showing 'Pump On', and a page indicator '1 of 1'.



The 'Local Event - Pump Off' dialog box is shown. It has a title bar with a question mark and close button. The 'Event activation:' section contains a 'Trigger:' dropdown menu with an 'Add' button. Below it is a list box containing 'Tank Level Full'. The 'Register write action:' section contains a 'Start register:' dropdown menu with 'Pump1Call (1001)' selected. Below this are 'Block size:' and 'Value:' fields, both containing the number '0'. On the right side, there are buttons for 'OK', 'Cancel', 'New', 'Copy', 'Delete', 'Notes', and 'Help'. At the bottom, there are navigation arrows, a 'Rename' button, a dropdown menu showing 'Pump Off', and a page indicator '2 of 2'.

A second Local Event can be used to turn OFF the pump when the signal from the same sensor goes above a setpoint:

Local Events Reference

The Local Event defines a register write action that is executed whenever one or more associated Triggers are activated. Each time the Trigger(s) activate, the specified Value will be written to the register block defined by the Start Register and Block Size. This feature can be used to implement simple control logic (such as: "when the analog input rises above the setpoint level, turn on the output").

Trigger

This allows you to select the Triggers that cause the Local Event to be activated. Individually select each desired Trigger or create a new one from New option in the drop down tree, then click the Add button.

Add Button

The Add button causes the selected Trigger to be added to the list of Triggers that activate the Local Event.

Start Register

The Start Register defines the starting point of the block of registers that will be written when the Local Event is activated.

Value

The Value will be written to all the registers in the defined register block when the Local Event is activated.

Block Size

The Block Size determines how many contiguous registers will be written with the specified value when the Local Event is activated.

This option is great for initializing large block of register to zero for example.

SECTION XII

Using Alarms



See *The ScadaBuilder Hierarchy* (on page 73)

Using ScadaBuilder, ICL controllers have built-in alarm handling without special programming. An Alarm can be used to directly operate an indicator light (Flash, ON, OFF), make a log file entry, and/or initiate a dialed out voice alert or pager message. Alarms may be viewed directly through the TUI using a TUI Alarm as well.



Alarms are particularly useful when used in conjunction with ScadaBuilder Dialers (requires voice modem option).

In most situations, Alarms have three states:

- No alarm
- Alarm active - not acknowledged
- Alarm active - acknowledged

ScadaBuilder keeps track of these alarm states to support alarm logging as well as supporting a real-time Alarm status display on a TUI HMI.

Alarms use Triggers defined in the ScadaBuilder Trigger section. These Triggers are used to initiate and acknowledge Alarms. See *Alarm Suitable Triggers* (on page 146) for details.



Alarm Triggers

For proper operation, Alarm triggers must be STATE, ABOVE or BELOW type Triggers.



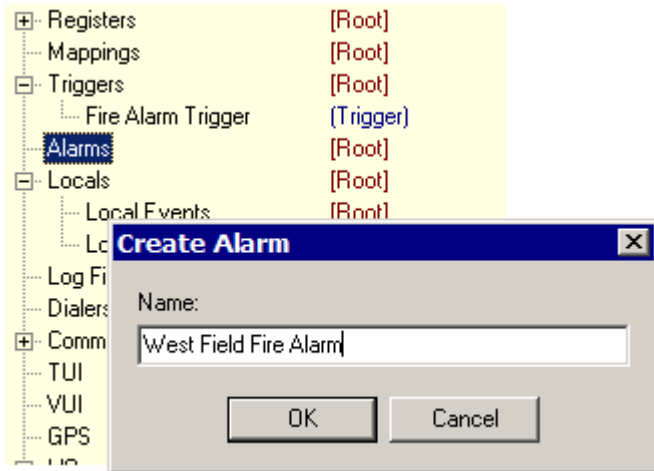
Alarm Acknowledgment Triggers can be any type such as an edge trigger for an acknowledge push button switch.

In This Section

Creating An Alarm	159
Defining An Alarm	160
Alarm Options	161
Alarms Reference.....	163

Creating An Alarm

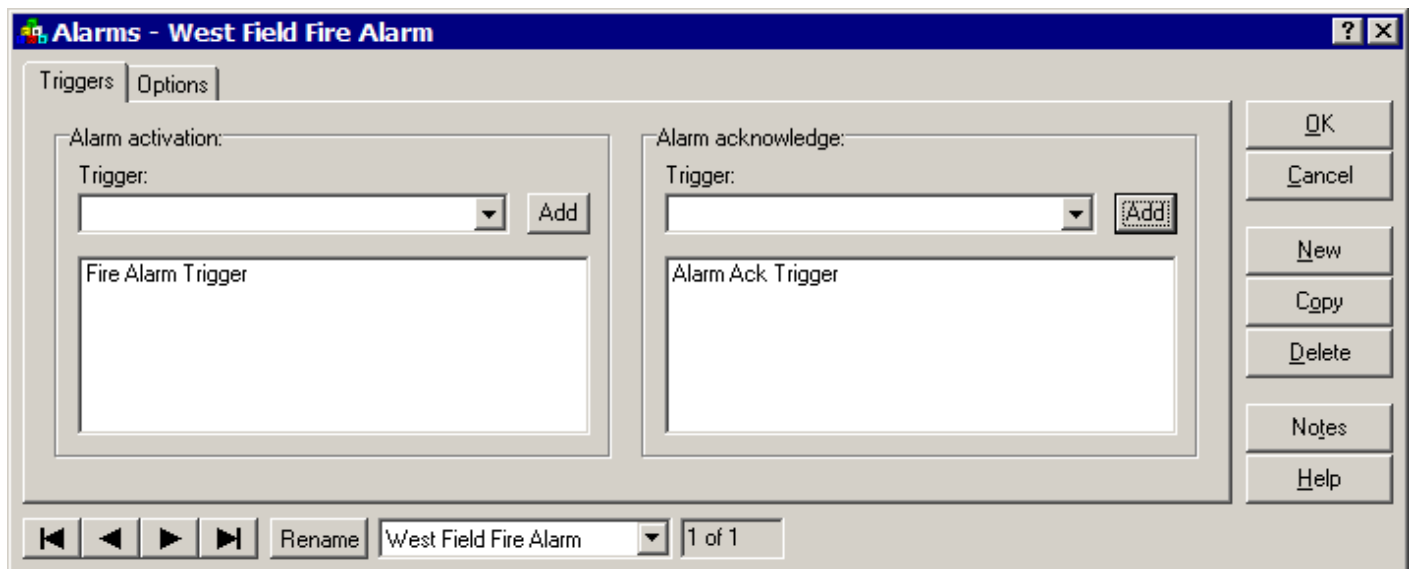
To create an Alarm, double-click on "Alarms" in the Project Manager (if there are no other alarms defined) or click on "NEW" in the Alarms window. A dialogue window will pop up for naming the Alarm. Accept the default name or enter a new one.



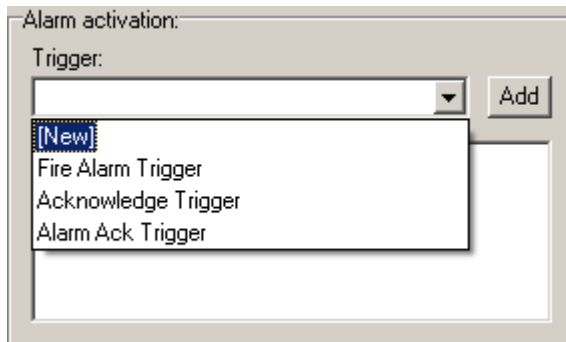
Once an Alarm is named, a window for defining the alarm is displayed.

Defining An Alarm

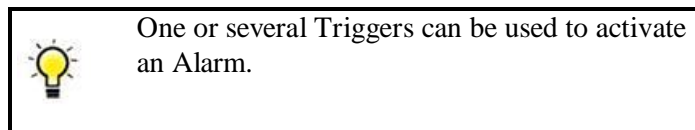
An Alarm requires two or more triggers:



Click on the Trigger drop down and select a Trigger from the list or create a new one by using the "New" option in the drop down menu tree:



After creating or selecting the Trigger in the drop down box, Click "Add" to add it to the list.

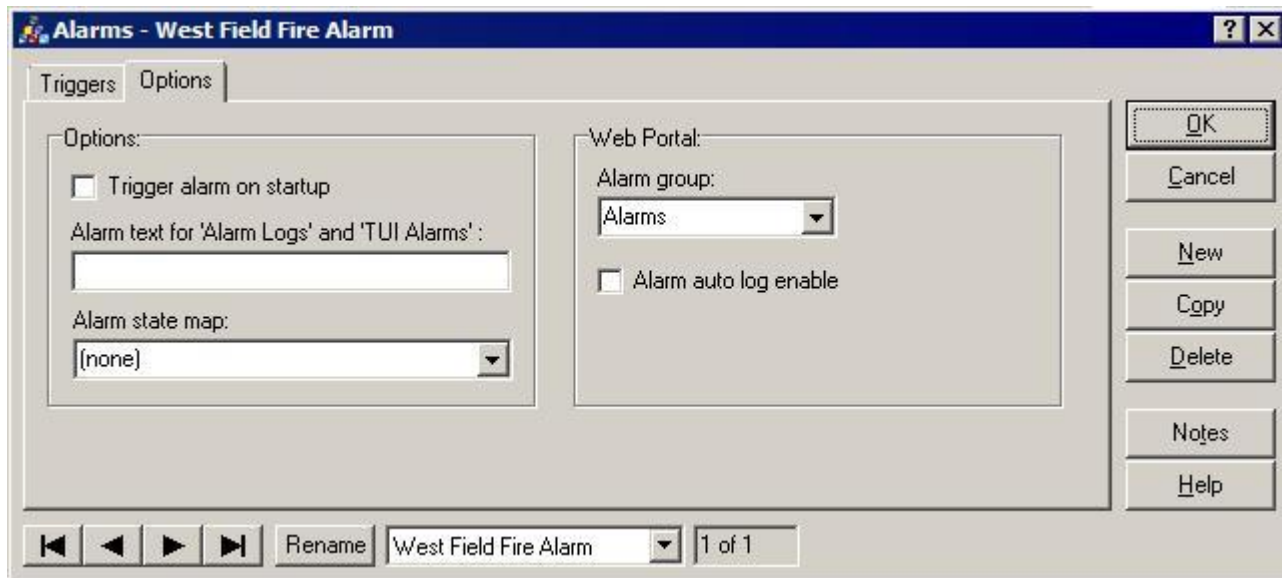


Likewise, one or more Triggers may be used to acknowledge an alarm.

A classic application is for one or more Triggers to be defined to monitor sensor inputs, while a second set of Triggers comes from a push-button switch and/or a Boolean point to be set by calling in with a touch-tone telephone, to clear the alarm.

Alarm Options

Clicking on the "Options" tab displays a new window for defining Alarm options:



Startup: If this box is checked, the controller will force this Alarm to occur on startup.

Alarm Log/TUI Alarm Name: This is an optional parameter that may be used to identify the Alarm for logging to a file and annunciation on a TUI (Text User Interface).

Alarm State Map. This allows the user to map the Alarm state to an Integer register. The integer will be set to the respective values:

0 = No alarm

1 = Alarm active - not acknowledged

2 = Alarm active - acknowledged

Web Portal

You can configure an Alarm to show up in the User Portal Interface by giving it an Alarm Group. If you would like to view and download reports for that alarm or alarm group, then check the Alarm Auto Log Enable checkbox.

Alarms Reference

Alarms are used for the detection, annunciation and logging of unusual ("alarm") conditions.

At any given time, an Alarm is in one of three states:

State	Description
Unacknowledged	The Alarm condition has occurred and the Alarm has not yet been acknowledged.
Acknowledged	The Alarm condition is still active and the Alarm has been acknowledged.
Idle	The Alarm condition is not active and any previous alarm condition has been acknowledged.

One or more Triggers define the conditions under which an Alarm is activated. One or more Triggers may be used to acknowledge an Alarm.

For instance, an Alarm may be activated from a Trigger when the value of an analog input connected to a pressure transducer rises above a threshold.

The Acknowledge function can be tied to a Trigger which is programed to respond to a digital input wired to a push-button

The Alarm state can be annunciated on a lamp tied to a digital output which is configured using a Local Alarm so that the lamp flashes for the Unacknowledged state, is on steady for the Acknowledged state and is off for the Idle state.

Activation Trigger

The Activation Trigger selection list allows you to select a Trigger that is to be added to the list of Triggers that cause the Alarm to be activated. Select the desired Trigger, then click the Add button.

Acknowledge Trigger

The Acknowledge Trigger selection list allows you to select a Trigger that is to be added to the list of Triggers that cause the Alarm to be acknowledged. Select the desired Trigger, then click the Add button.

Add Button

The Add button causes the selected Trigger to be added to the list of Triggers.

Option Trigger on Startup

If the Trigger on Startup option is selected, the Alarm will be automatically triggered on application startup.

Alarm Text

This is an optional parameter that may be used to identify the Alarm for logging to a file and annunciation on a TUI (Textual User Interface).

State Map

Alarm - State Map

Maps an Integer register to the alarm to read the state of the alarm.

Alarm State Values:

- 0 - Alarm Idle
- 1 - Alarm Active
- 2 - Alarm Acknowledged

Alarm Group

Used by the Web Portal to organize and acknowledge alarms by groups in the User Portal interface.

Alarm Auto Log Enable

Check this box if you want reports for this Alarm and Alarm Group in the User Portal.

Using Local Alarms



See *The ScadaBuilder Hierarchy* (on page 73)

Local Alarms are a way of annunciating Alarms to local I/O or registers.

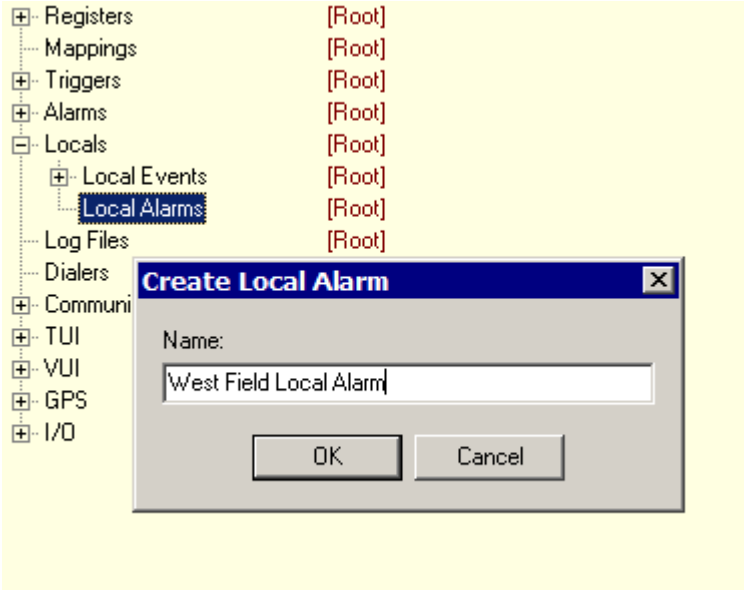
In This Section

Creating a Local Alarm	165
Defining a Local Alarm	165
Local Alarms Reference	166

Creating a Local Alarm

A Local Alarm is used by an Alarm to annunciate to local I/O. A Local Alarm may be setup to flash during an alarm condition for a lamp or beacon. It may be set up to just turn on a digital output in the alarm state for running a horn or sonic device.

To create a Local Alarm, click on “Local Alarms”. A window will pop up to name the local alarm. Accept the default name, or enter a new one.



Defining a Local Alarm

Once the local alarm is named, a window for defining the alarm action is displayed:

A Local Alarm requires one or more Alarms (see previous section on Alarms).

Here, an alarm from a group of discrete sensors has been defined (Westfield Fire Alarm). The local alarm configuration includes a register for alarm annunciation and definitions for the actions to be taken for each of three alarm states; In alarm - Unacknowledged, In alarm - Acknowledged, and no alarm - Idle . In this example, the Discrete Output is presumably driving an indicator light. When the alarm is active and not acknowledged, the light will flash ON and OFF twice per second (0.5 seconds ON, 0.5 seconds OFF). When the Alarm is acknowledged but the alarm condition still exists, the light will remain ON solid.

When the Alarm is clear, the light will be turned OFF. Note that it is fairly simple to create other alarm sequences with variations on flashing rate by changing these settings.



Local Alarms are typically used to flash a light indicating a Common Alarm. This way, any active alarm will make the DO it is assigned to flash until acknowledged.

Common Alarms may also use an ON state to sound an audible alarm via a horn or other sonic device. The Ack and Idle state would then be set to OFF.

Local Alarms Reference

A Local Alarm controls the "local" annunciation of an Alarm. Local annunciation is normally performed with a lamp or buzzer that is attached to a digital output on the controller. A typical setup would control a digital output like this:

State	Action
Unacknowledged	toggle
Acknowledged	on steady
Idle	off

Alarm

The Alarm selection list allows you to choose an Alarm that is to be added to the list of Alarms that cause the Local Alarm to be activated. Select the desired Alarm, then click the Add button.

If multiple Alarms are listed, the Local Alarm annunciates the "worst case" state of all the alarms. If any Alarm is active/unacknowledged then the "Unack Mode" annunciation is applied. If no Alarms are unacknowledged, but at least one Alarm condition is in the Acknowledge state, the "Ack Mode" annunciation is applied. Only if all Alarm conditions are inactive is the "Idle Mode" annunciation applied.

Alarm Annunciation Register

The Alarm Annunciation Register is the register that gets written to automatically annunciate the Alarm based on its state. Typically the register is mapped to a digital output.

Unack Mode

The Unack Mode determines how the Alarm Annunciation Register is controlled when the associated Alarm is in an Unacknowledged state. The modes are as follows:

Mode	Action
Cycle	An on/off (1/0) value is alternately written to the register according to the configured On Time and Off Time.
On	An on (1) value is written to the register.
Off	An off (0) value is written to the register.

On/Off Time

The On Time and Off Time set the frequency and duty cycle when the register value is being toggled between on (1) and off (0). The times are given in milliseconds. The default values (500ms/500ms) give you a 1 Hz frequency and a 50% duty cycle.

Ack Mode

The Ack Mode determines how the Alarm Annunciation Register is controlled when the associated Alarm is in an Acknowledged state. The modes are as follows:

Mode	Action
Cycle	An on/off (1/0) value is alternately written to the register according to the configured On Time and Off Time.
On	An on (1) value is written to the register.
Off	An off (0) value is written to the register.

Idle Mode

The Idle Mode determines how the Alarm Annunciation Register is controlled when the associated Alarm is in an Idle (inactive) state. The modes are as follows:

Mode	Action
Cycle	An on/off (1/0) value is alternately written to the register according to the configured On Time and Off Time.
On	An on (1) value is written to the register.
Off	An off (0) value is written to the register.

Using Alarm Dialers

Alarm DialerFILE2258

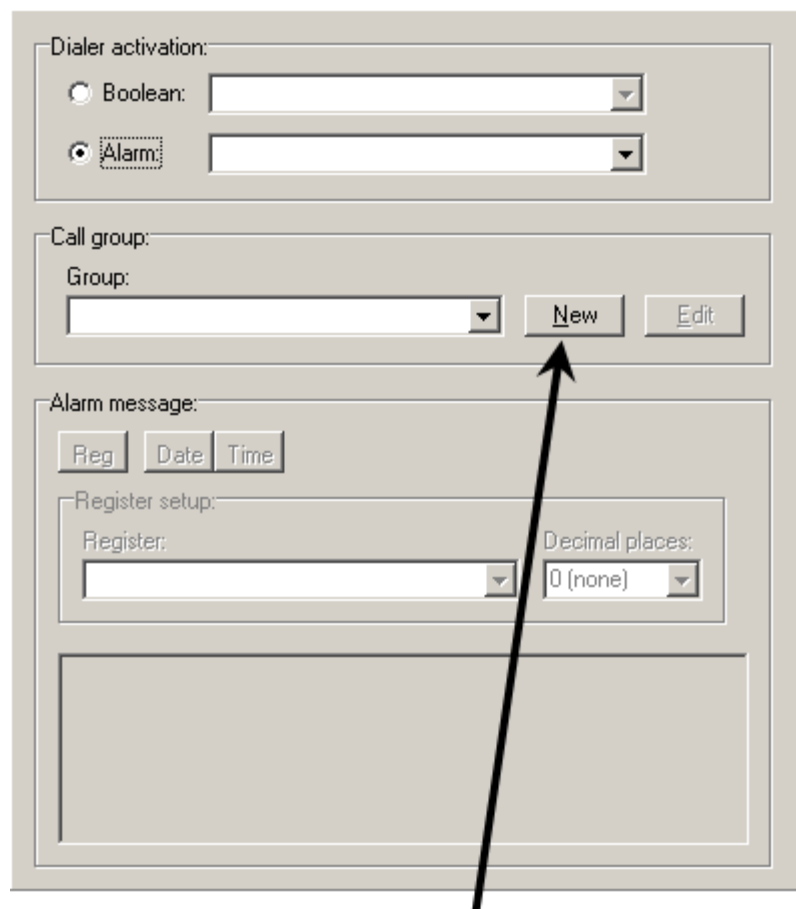
In This Section

- Creating an Alarm Dialer..... 169
- Setting Up a Call Group 170
- Finishing the Alarm Dialer Configuration 172
- Alarm Dialer Reference..... 173
- Call Group Reference..... 175
- Generating and Downloading a Voice File 178

Creating an Alarm Dialer

Creating an Alarm Dialer

To create an Alarm Dialer, double-click on “Dialer” in the Project Manager (if there are no other Dialers defined) or click on “NEW” in the Alarm Dialer window. A dialogue window will pop up for naming the Alarm Dialer. Accept the default name or enter a new one.



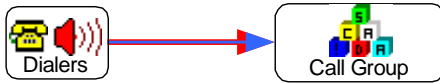
Create a new call group by clicking on this button.

Once the Alarm Dialer is named, a window for defining the dialer's operation is displayed. An Alarm Dialer requires a boolean point or an Alarm to activate the alarm dial out. You can create a new boolean or Alarm record or use existing ones in the drop down boxes.

You will also need a “Call Group”. If you don't already have one defined, you can create one from this window.

See Using a Call Group section and Using Alarm Dialers (Continued) section.

Setting Up a Call Group



See *The ScadaBuilder Hierarchy* (on page 73).

You must have a Call Group in order to configure an Alarm Dialer. If you don't have one already, you can click on the "New" button in the Call Group section of the configuration window in order to set one up.

Your Call Group should look something like this:

Select your protocol

Type an introduction message

Select an acknowledge code

Go to the Dialing tab to enter phone number list.

Setup Dialing

Phone number list:

Number/Buffer:

Add

555 623-8123
555 613-4567

Alarm annunciation retries:

Wait 0 minutes before dialing the next phone number in the list.

Retry a phone number 0 times before moving on to the next number.

Wait 1 minutes before repeating the phone number list.

Retry the phone number list 2 times.

Enter your phone number list click Add for each one

Click OK to return to the Alarm Dialer dialog



Please see the Call Group Reference (on page 175) in the following section for more information on configuration parameters

Finishing the Alarm Dialer Configuration

The Alarm Dialer can be activated from either a Boolean point or from an Alarm. The Boolean point can be either a “physical” I/O point or an internal “coil” driven by programmed logic.

The Alarm message is the actual message to be played or displayed. This can be one or more lines of date, time, text and register data. Enter the date and time by clicking on their respective buttons.

Enter register data by selecting a register tag name and clicking on the “Register” button”.

Enter text “free form” just like a normal text editor.



Placing quotation marks around a block of text tells ScadaBuilder that it should try to reuse this same phrase if it appears in any other messages used in this Node. Creating alarm messages with common phrases can significantly reduce memory usage at the expense of some quality. Breaking the message up into phrases does not sound quite as nice as one continuous sentence because of the computer generated inflection placed on certain words in the sentence.

Once you hit OK and your Dialers are done. Generate the voice file and download it to the controllers. See Generating A Voice File section.

Alarm Dialer Reference

An Alarm Dialer provides a way to annunciate alarms over a phone line by either generating spoken voice messages or sending text to numeric and alphanumeric pagers.

Call Group

Specifies the name of the Call Group that the dialer is associated with. The Call Group is used to specify what communication and dialing parameters are used when the dialer is activated.

See the Call Group section for more detailed information.

New Button

The "New" button allows you create a new Call Group that will then be added to the list.

Edit Button

The "Edit" button allows you to modify the Call Group that is currently selected.

Dialer Activation

This allows you to select either a Boolean register or an alarm that will be used to activate the alarm dialer.

If a Boolean register is selected, the alarm dialer will dial out to annunciate the specified message when the register transitions from a 0 to 1. The register must return to a 0 value for one scan loop before it can be triggered again.

If an alarm is selected, the alarm dialer will dial out to annunciate the specified message when the alarm is activated. Note that an alarm and associated trigger must first be created before selecting this option.

Alarm Message

The Alarm Message specifies the message that is played or displayed when the dialer is activated. The value of one or more registers along with the date and time can be inserted into alarm message (depending on the protocol selected in the Call Group).

Register

Selects the register for which the value will be inserted into the alarm message when the "Register" button is clicked.

Decimal Places

Specifies the number of decimal places that will be annunciated for a floating point register when inserted into the alarm message with the "Register" button.

Register Button

The "Register" button inserts the register specified in the "Register Setup" box into the alarm message. The current value of the register will be annunciated when the dialer is activated.

Date Button

The "Date" button inserts the date into the alarm message. The date (month/day) when the dialer was activated will be annunciated in the message.

Time Button

The "Time" button inserts the time into the alarm message. The time (hour/min) when the dialer was activated will be annunciated in the message.

Preview Button

Click the preview button to listen to the message entered into the alarm message box. Registers, date and time values will be annunciated as their token names versus the actual values which are heard when the application is running. Right-click the preview button to bring up the voice settings dialog.

Settings Button

Specify the voice and playback speed of the Alarm Dialer. See the *Generate Voice Reference* (on page 182) section for more details.

Call Group Reference

A Call Group allows you to specify the dialing parameters that are used with an Alarm Dialer. You can specify parameters such as the protocol (voice/pager), Modem port and phone number list.

NOTE: A Call Group is a specialized Network Session that is tailored especially for Alarm Dialers. You can also access a Call Group through the Network Session dialog. This allows you to modify the more advanced communication parameters if so desired. Note that you will not be able to change the protocol when accessing a Call Group in the Network Session dialog.

Setup Tab

Group Name (Session)

Specifies the name that will be used to reference the Call Group.

NOTE: This name will also show up in the list of Network Sessions. A Call Group can also be modified through the Network Session, providing access to the more advanced communication parameters.

Protocol

Defines the protocol used for this Call Group.

Alphanumeric Pager (TAP) This protocol supports sending messages to alphanumeric pagers.

Numeric Pager This protocol supports sending messages to numeric pagers.

Voice This protocol supports sending voice messages to a telephone.

Switch To VUI

Selecting a Voice User Interface (VUI) entry will make the Call Group automatically switch to the specified VUI after the user exits the Alarm Dialer. This only applies to voice alarm Dialer types.

Modem Port

Selects the communications port (Network Port) that is used to deliver alarm messages specified by the associated alarm dialer configurations.

If the controller model supports an internal modem then a "(default)" selection will appear for the communications port. This selection will use the internal modem with pre-determined communication. To use a different port or change a pre-determined parameter you will first have to create Network Port with dialup enabled. You can then select this as the communications port in this dialog.

Note: A controller must have a Modem option configured to use this feature. See *Node Settings and the General tab* (see "Node Settings - General Tab" on page 34) for more information.

Introduction Message

The introduction message specifies the message that will be played or displayed before the list of alarm messages. This is typically used to identify the unit that is sending the message.

Preview Button

The preview button plays the text specified in introduction message. This is used to sample the voice output with the current voice settings before downloading the configuration to the node.

Right-click the preview button to bring up the voice settings dialog.

Settings Button

In the Setting dialog you can set all configurable parameters of the voice you are going to use. See *Generate Voice Reference* (on page 182) for more details.

Alarm List Repeat Delay

The alarm list repeat delay specifies how much time to wait before the list of alarms is repeated.

This parameter only applies to the Voice protocol.

Also labeled "Wait x milliseconds before repeating Alarm Message List."

Alarm List Repeat Count

The alarm list repeat count specifies how many times the list of alarms is repeated before hanging up. To terminate the call sooner and let the controller know that you are hanging up, press the '*' key on the keypad. When the controller hangs up you will here the word "Goodbye" played.

This parameter only applies to the Voice protocol.

Also labeled "Repeat the Alarm list x times before ending call."

Status Buffer

The Status Buffer is used to display and record diagnostic information from the Call Group's activity. It will display such statuses as "Dialing", "Greeting", "Alarming", "Acknowledged" and "Goodbye" to help track what state the alarm

Dialer and Call Group are currently working on.

Alarm Acknowledge String

String Constant

When a string constant is entered, the Alarm acknowledge string specifies the key sequence you must enter from your keypad to acknowledge the active alarms. Any alpha-numeric value may be used in the sequence.

Buffer or Message Register

When a buffer or message register is selected, the alarm acknowledge buffer holds the key sequence you must enter from your keypad to acknowledge the active alarms.

After the sequence is entered on your keypad you must always follow it by the '#' key. If no sequence is specified then pressing the '#' key by itself will acknowledge the alarms.

This parameter only applies to the Voice protocol.

Dialing Tab

Phone Number

The telephone number specifies a specific number to add to the phone number list box when the associated "Add" button is clicked.

Phone Number Buffer

The telephone buffer specifies a buffer register to add to the phone number list box when the associated "Add" button is clicked.

Using a buffer instead of "hard coding" a number allows you to change the telephone number while the application is running.



For formatting integer registers into phone numbers, see **Buffer Format Editor** (on page 126) help and **Options** (on page 127) for details.

Redial Wait

Specify the time to wait between phone numbers when a connection was NOT established.

Connections for Dialers are qualified by the Ring Go Away Time configured in the Modem (Dialup) section of the Network Port applied to this Call Group.

The default for the Ring Go Away Time is 1 meaning it will always connect. In the United States, telephone rings are 5 seconds apart so the RGA will have to be 6 seconds in order to determine a connection. This often results in a pregnant pause after the receiver is off hook. (It sounds like a telemarketer is waiting to connect with you).

Redial Retry Count

Number of retries (after the first try) to connect to the current number.

Number Wait

Specify the number of minutes to wait before the next number in the list or the current number is to be called. This parameter is also known as "Wait x minutes before dialing the next number in the list."

Number Retry Count

Specify the number of times after the first call to retry the current number in the list. The default for this parameter is 0 so that the list will run through calling each number in the list before repeating the list.

This parameter is also known as "Retry a phone number x times before moving to the next number."

List Wait

Specify the number of minutes to wait before sequencing through the list of phone numbers again.

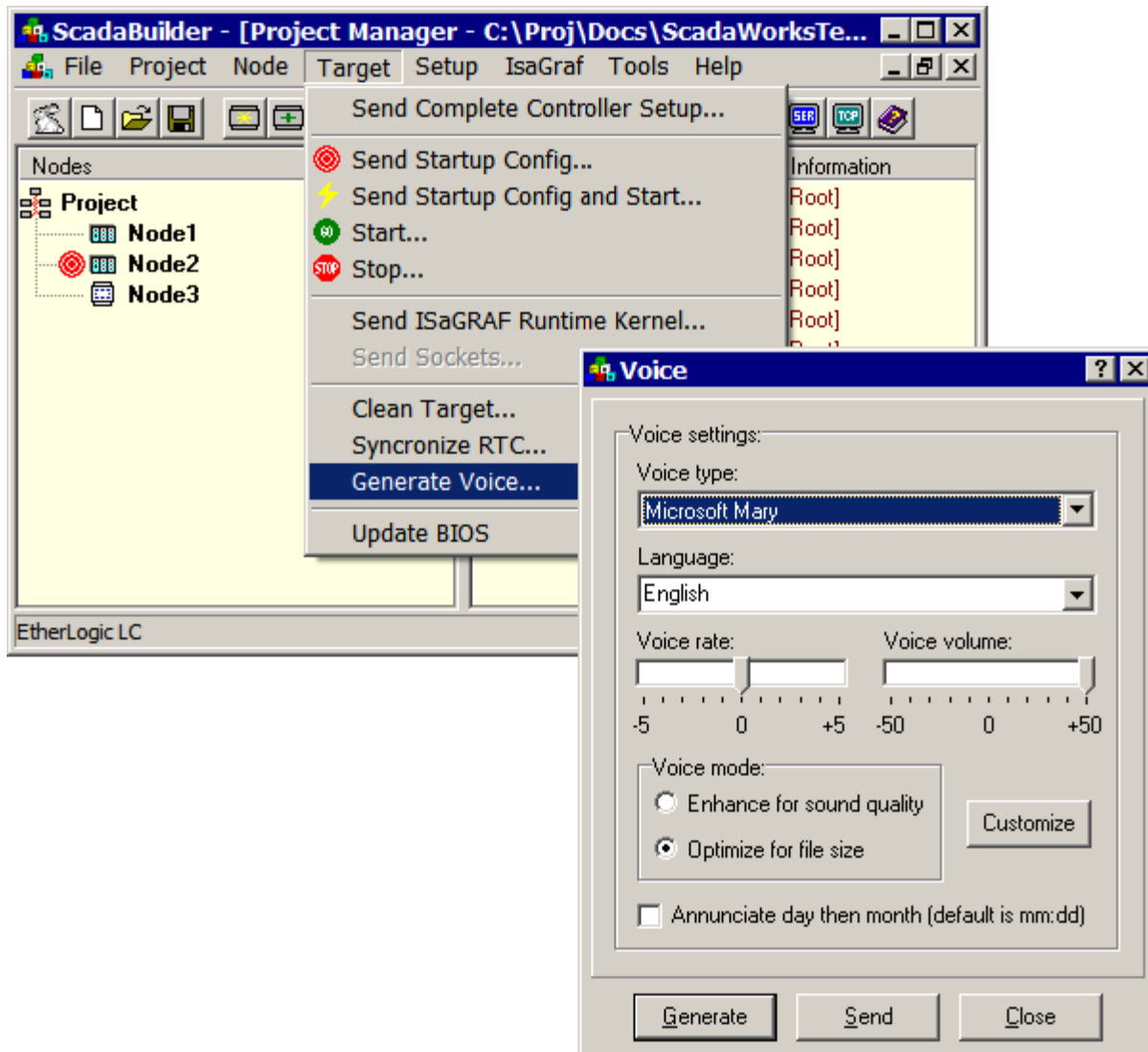
Also known as "Wait x number of minutes before repeating the phone number list."

List Retry Count

Specify the number of times to repeat the phone number list after the initial run through the list. System will continue dialing until the alarm is acknowledged or all retries are exhausted.

Generating and Downloading a Voice File

If you are using voice alarm Dialers or the Voice User Interface (VUI) , you must generate and download a voice file to the controller whenever a voice message has been added or changed. This is done by selecting **Target | Generate Voice...** menu in the Project Manager. You must also do a “Make” and program download to fully implement the changes.



*Doing a **Target | Complete Controller Setup...** is often the best way to get things working if you have problems with messages coming out garbled.*

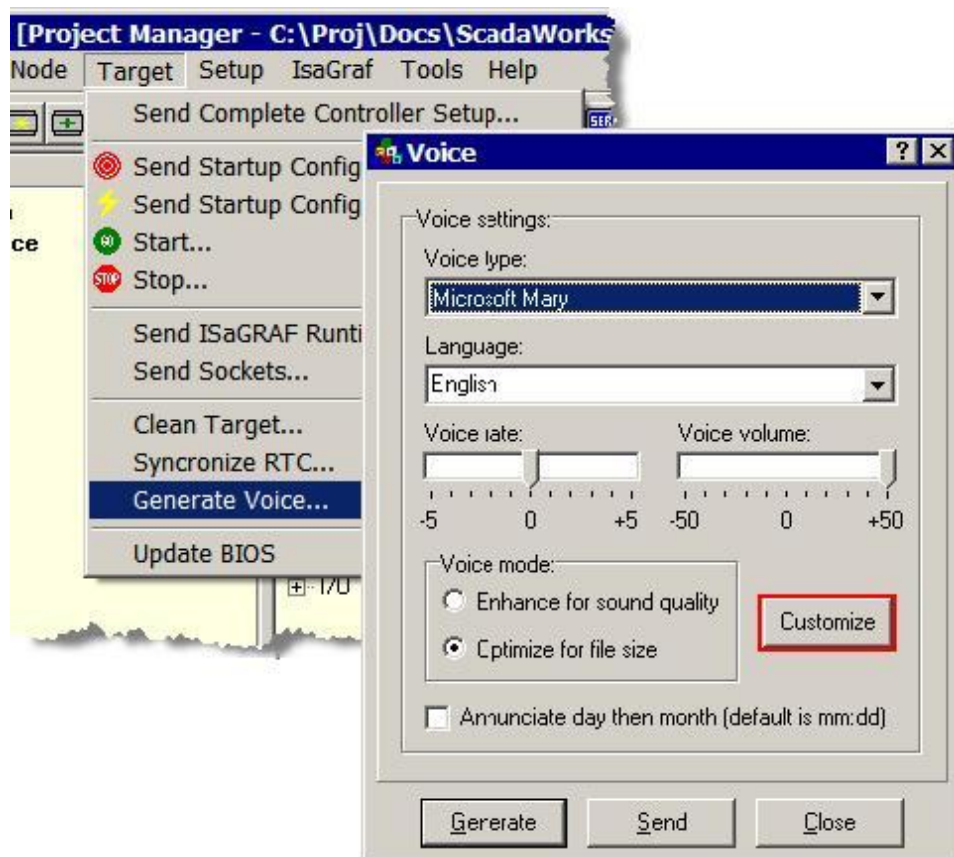


Voices that start LM_ or end with _LM will not work with ScadaWorks. These voices are used on several different competitive products and may show up in the Voice type drop down box.

You must have a compatible sound card and the sound device must be enabled for the voice to file conversion to work properly.

Click on the Send button to transfer the voice file to the Controller. ScadaBuilder will use FTP or Zmodem serial, whatever it is configured for in the **Node | Settings FTP/HTTP** tab.

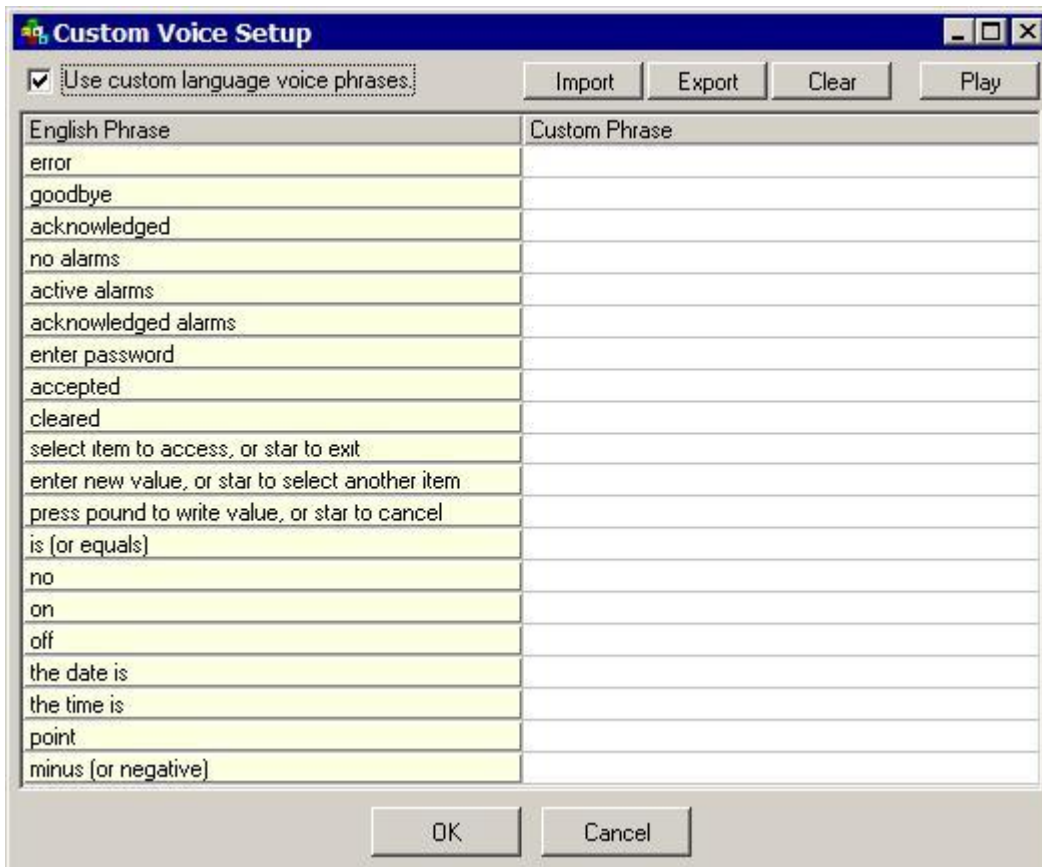
Customizing the Voice Interface (English and Non-English)



The ICL Controller voice system allows you to customize your own phrases to suit your needs. Customizing is essential when support of any one of the non-English languages is involved. You can select the voice you want and the language you want as well. Then click on the Customize button when you are ready to define your custom messages.



Select your voice and your language here and click on Customize.



These are the Voice User Interface and Alarm Dialer default voice messages that get translated into an 8bit, 8KHz PCM voice file.

You must check the "Use custom language voice phrases" checkbox.

Enter your phrase for each function here. Use the Export button to save the data for reuse and the Import button to recall it.

When you are done, do a **Target | Complete Controller Setup...** to make sure the program is in sync with the voice file.



When using a non-English language, you must redefine all messages for the interface to generate properly.

Generate Voice Reference

The Generate Voice dialog is used to generate and update the voice wave file (voice.sbw) on the target controller.

IMPORTANT: The node files should always be resent to the target after the voice files have been regenerated. This will re-sync the application voice file message indexes.

Voice Type

The voice type specifies the voice that is used when generating the voice file. Default voices are installed as part of the Windows Sound API. Third party voices, that may offer more clarity, may also be installed and then selected from the list.

Voice Rate

The voice rate specifies the rate at which speech is rendered. As the rate is decreased the size of the voice file will increase.

Volume

Allows setting the gain at which the voice wave file is recorded. The volume will be louder over a phone handset for example.

Voice Mode

The voice mode will effect the quality of the speech rendered. Selecting "Enhance for sound" will individually render all numeric value samples, increasing the size of the voice file. Selecting "Optimize for file size" will only render a subset of the numeric value samples, decreasing the size of the voice file.

Generate Button

The "Generate" button will generate the voice file with the specified settings and send it to the target.

IMPORTANT: The node files should always be resent to the target after the voice files have been regenerated. This will re-sync the application voice file message indexes.

Send Button

The "Send" button will send a previously generated voice file to the target without generating a new file.

Language

Scadabuilder also allows other languages besides English including French, German, Spanish and Portuguese.

For a list of supported vendors, contact ICL at support@iclinks.com or (US) 530-888-1800

Play Button

The "Play" button will allow the message to be previewed with specified voice type and rate.

Using the Voice User Interface



See *The ScadaBuilder Hierarchy* (on page 73)

ICL controllers purchased with an optional internal telephone modem can support a Voice User Interface (VUI). This feature enables worldwide access to information in the controller using a standard touch-tone telephone. After validating access with a password, an authorized user may examine and optionally change values in controller registers. The user interacts with the controller by entering information through a touch-tone keypad and listening to controller vocalized messages. The controller messages consist of canned descriptive text along with register data.

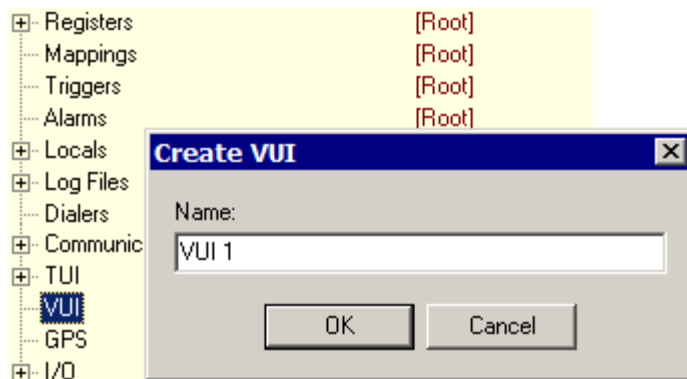
VUI messages are synthesized based on register selections and text that is entered into the VUI configuration. A choice of several male and female voices are provided. The voice phrases are stored in a data file on the controller's flash disk and maybe easily cloned to replicate to other controllers. Each minute of voice storage requires approximately 0.5MB of disk space. Controllers can be configured to reuse phrases that are common between messages significantly reducing the storage required.

A VUI may be linked to an Alarm Dialer so that once the alarms have been acknowledged, the person called can interact with the controller to gain more information related to the alarm. The registers enabled for access and the access passwords are set up in the ScadaBuilder VUI section.

In This Section

Creating a Voice User Interface	183
Controlling VUI Access	187

Creating a Voice User Interface



To create a VUI, double-click on "VUI" in the Project Manager (if there are no other VUI's defined) or click on "NEW" in the VUI window. A dialogue window will pop up for naming the VUI. Accept the default name or enter a new one.

Click OK.

You should see a screen like this. Configure the menu items to your liking. The codes are arbitrary and can be assigned by you.

Setup | Login | Misc

Access setup:

Network port: Internal Modem Status buffer: (none)

Greeting message: This is the ICL Alarming System Preview Settings

Access prompt:

Prompt repeat time: 3000 (ms)

Prompt retry count: 2

Access entry:

Add Access code: Access name: Register: Decimal places: 0 (none) Units: Read only

Add Access code: 5 Call group: Operator Group Alarm state: both

Register access entries:

Code	Name	Register	Decimal	Units	RO
1	Pump 1 Running	DI1			x
1	Pump 2 Running	DI2			x
2	Tank Level	AI1		Feet	x
3	Pump On Level	AO1		Feet	

Alarm access entries:

Code	Call Group	State
5	Operator Group	both

Move Up	Ctrl+Up
Move Down	Ctrl+Down
Delete	Del
Preview Name	Ctrl+N
Preview Units	Ctrl+U
Edit Code	Alt+C
Edit Name	Alt+N
Edit Units	Alt+U
Read Only	

Right-clicking on an Access List entry displays a pop up menu that contains editing options for the entries in the list. The entries may be moved up or down in the list or deleted. Associated message fragments for each the entry may be previewed. The fragments may also be edited.

Setup Tab Reference

Primary configuration tab for the VUI.

Network Port

Identifies which Network Port to use for the Voice User Interface. This port must be configured as a Modem and use an ICL modem model for the voice system to work. A standard US Robotics Sportster for example, will NOT work.

Status Buffer

Identifies a buffer used to show protocol status for debugging/monitoring purposes. Currently reports modem dialing and voice playback status information.

Greeting Message

The greeting message specifies the message that will be played when a user dials in the VUI. This is typically used to identify the unit that is running the VUI.

Preview Button

The preview button plays the text specified in greeting message control. This is used to sample the voice output with the current voice settings before downloading the configuration to the node. Right-click the preview button to bring up the voice settings dialog.

Settings Button

Setup voice and speech rate to use. See *Generating and Downloading a Voice File* (on page 178) for more details.

Prompt Repeat Time

The prompt repeat time specifies how much time to wait before a voice prompt is repeated.

Prompt Retry Count

The prompt retry count specifies how many times a voice prompt is repeated. When repeat count has been reached, the VUI will move back to the previous prompt level or hang-up if already at the top level. To terminate the call sooner and let the controller know that you are hanging up, press the '*' key on the keypad. When the controller hangs up, you will hear the word "Good bye" played.

Read/Write Password String

When the "String" radio button is selected, the read/write password string specifies the key sequence you must enter to gain full access to the VUI system. Any alpha-numeric value may be used in the sequence. After the sequence is entered on your keypad, you must always follow it by a '#' key.

NOTE: If neither a read/write nor read-only sequence is specified then the user will automatically have full access to the VUI system without being prompted for a password.

Read/Write Password Buffer

When the "Buffer" radio button is selected, the read/write password buffer holds the key sequence you must enter to gain full access to the VUI system. Any alpha-numeric value may be used in the sequence. After the sequence is entered on your keypad, you must always follow it by a '#' key.

NOTE: If neither a read/write nor read-only sequence is specified then the user will automatically have full access to the VUI system without being prompted for a password.

Read Only Password String

When the "String" radio button is selected, the read-only password string specifies the key sequence you must enter to gain read-only access to the VUI system. Any alpha-numeric value may be used in the sequence. After the sequence is entered on your keypad, you must always follow it by a '#' key.

NOTE: If neither a read/write nor read-only sequence is specified then the user will automatically have full access to the VUI system without being prompted for a password.

Read Only Password Buffer

When the "Buffer" radio button is selected, the read-only password buffer holds the key sequence you must enter to gain read-only access to the VUI system. Any alpha-numeric value may be used in the sequence. After the sequence is entered on your keypad, you must always follow it by a '#' key.

NOTE: If neither a read/write or read-only sequence is specified then the user will automatically have full access to the VUI system without being prompted for a password.

Add Button

The Add button causes the access entry to be added to the access list or the Alarm list.

Access Code

The access code specifies the key sequence that will be used to access the associated register through the VUI system. When the VUI prompts you to select a register to access, key in the code for the desired register followed by a '#' key. The VUI will not allow duplicate sequences that map out to be equivalent strokes on the key pad.

Access Name

The access name specifies the voice message that is played back to the user after an access code has been entered. This allows the user to verify the register that has been selected.

Units

The unit specifies the voice message that will be played after the value of the associated register has been played. This is typically is used to play messages such as "feet", "PSI", "degrees C". etc.

Register

Selects the register that is to be accessed in the VUI when the associated access code is entered by the user.

Decimal Places

Specifies the number of decimal places that will be annunciated for the associated register when accessed in the VUI. Decimal places are only selectable with floating-point type registers.

Call Group

Selects the Call Group (setup in the Dialer dialog) from which to draw acknowledged or unacknowledged Alarms.

Alarm State

Selects whether to play acknowledged, unacknowledged or both types of Alarms from the given key code.

Controlling VUI Access

The VUI allows as many security accounts as you like for controlling access. Read/write and read-only accounts can be set up so that operators or unqualified personnel cannot access registers that they should not. This feature is accomplished with the Login tab of the VUI dialog.

Type	Password	Type	Username	RO
(string)	wfjhtxm	(string)	administrator	
(string)	uhvhbebe	(string)	operator	
(string)	user	(string)	user	x

Login Tab Reference

The tab contains all configurations concerning VUI security and user login accounts

User Login Buffer

This option selects a buffer (message) register to display the login activity of the current user. The message register may be applied to a Logfile Entry using a buffer Trigger to record who logged in or out and when.

Login Add

This button adds the currently configured Password and Username to the access list. The read only parameter is also stored.

Login Password

Configures the user's password for the current access entry.

Login Password Buffer

Allow a buffer or message register to be configured to hold a user's password. This allows account information to be changed at runtime.

Login Read Only

This flag sets the ability to restrict a particular user to reading values only. The user will not be able to modify register values when logged in.

Login User Name

Enter an access entry's user name here.

Login User Name Buffer

Allow a buffer or message register to be configured to hold a user's username. This allows account information to be changed at runtime.

Misc Tab Reference

Miscellaneous VUI parameters

Number of Rings Before Answering

System will wait this number of rings before picking up the phone line allowing after hours pick of the VUI on a shared voice line.

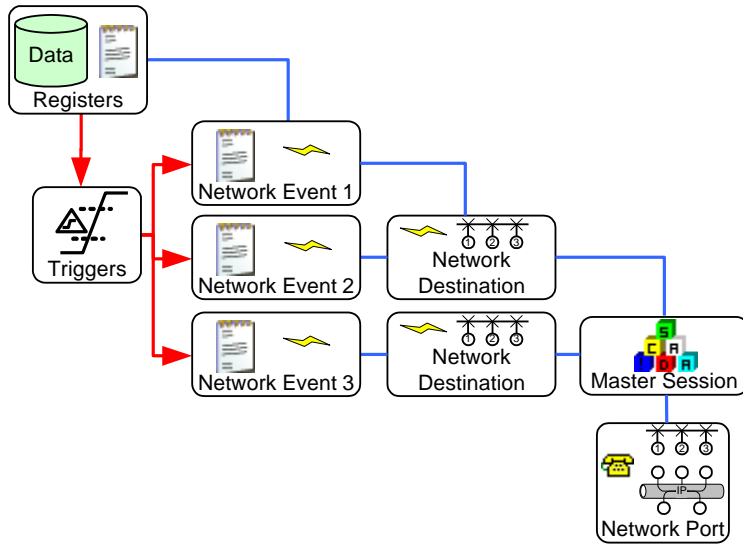
Default Monitor-Answer Session

Setting this checkbox identifies the Network Session as the default session that will run when no other Network Session is active. This allows it to monitor for incoming messages and dialup connections.

This parameter also applies to any VUI Network Session on a modem port.

Communications

Hardware, How, What and When



ICL controllers support a multitude of communications mediums including RS-232 and RS-485 serial ports; internal modems, radios as well as high-speed Ethernet.

See *The ScadaBuilder Hierarchy* (on page 73) for more information.

The ScadaBuilder Communications section is used to configure these hardware ports, define the protocol(s) to be used for communications through each port, and, when the Controller is to serve as a “Master” device, the data to be sent out or retrieved based on user defined events.

The configuration of a communications port can consist of up to four components:

Network Port (Hardware)

This component defines the hardware level parameters for communications such as baud rate and parity. If the port supports RS-485 operation, the timing control parameters are set in this section. If the port is to be used for peer-to-peer communications, the parameters that control message “collision” handling are defined here also. The Ethernet port is also considered a Network Port though its configuration is done in the **Node Settings - Ethernet / Serial IP Tab** (see "Node Settings - Ethernet Tab" on page 37) of ScadaBuilder.

Setup	Information
[-] Communications	[Root]
[-] Com1	(Network Port)
[-] Com2	(Network Port)
[-] Com5	(Network Port)
[-] Modem	(Network Port)
[-] Usb1	(Network Port)
[-] Usb2	(Network Port)
[-] STM Session	(Network Session)
[-] STM Destination	(Network Destination)
Read STM AI's	(Network Event)
[-] Ethernet	(Network Port)
[-] Modbus UDP Master	(Network Session)
[-] Ascent Dest	(Network Destination)
Read AI's	(Network Event)
Write DO's	(Network Event)



The Network Port is the HARDWARE that the controller uses to communicate with.

Network Session (How The Hardware Network Port Is Used)

This component defines one or more communication protocols to be used on a Network Port, as well as options for manipulating data formats being handled by the protocol. Most common communication protocols (such as Modbus) are set up to handle 16-bit values, with “modifications” to support 32-bit integer and floating point numbers as well as Message (buffer) registers. Since ScadaBuilder-ISaGRAF registers are 32-bits wide, the Network Session allows for the adjustment of the register data to match the protocol. This includes any required byte reordering to match various vendor’s “versions” of these protocols for handling 32-bit Integer, floating point values and strings.

The Network Session also has several options for monitoring communication link quality and keeping statistics to help troubleshoot communication problems. It also allows for the configuration of connection timeouts and retries as well as controlling probing (slow poll) modes when a communications failure occurs.

In many circumstances, it is possible to have multiple Network Sessions on a single Network Port.



The Network Session represents HOW the controller transfers data including protocol, timing, recovery and diagnostic statistics.

Network Destinations (Where The Data Will Be Transmitted To Or Retrieved From)

This component defines where the message will be transmitted to. A Network Destination is responsible for the address, IP address and phone number of the device being queried. It may also be configured with statistics, a comm fail flag and a boolean to disable this particular address.



The Network Destinations represent WHERE on a network the subsequent Network Events under it will get and retrieve their data.

Network Events (What Data Is Transmitted Over The Network Session)

This component defines messages to send or request data. It is used with “Master” or peer-to-peer protocols. There are typically multiple Network Events associated with a Network Session, each event sending or requesting a block of data.

All communications interact with registers in the controller. If the port serves as a “Slave” device, then the controller responds to requests for register data. If the requested registers are not defined in the controller, values of “0” are returned for each non-existent register. If the port serves as a “Master” device, then the registers used for transferring data must be defined before setting up the events that use those registers.

There are many kinds of Network Events including:

- Modbus, Bricknet, DNP3, DF1, Ethernet I/P and Hart register read and write Network Events
- Bricknet Probe Events (communications check)
- Email Events (with attachments if desired)
- FTP Events (sending and retrieving files)
- File Transfer Events (sending and retrieving files over Modbus and Bricknet serial networks)
- Real Time Clock read and write events (over Modbus or Bricknet serial networks)



The Network Event represents WHAT DATA the controller transfers over the Network Destination, Network Session and Network Port.

Triggers (When Will the Data Be Sent or Read)

While Network Events have their own mechanism for activating periodically, the Trigger represents a tool where the user can decide when the controller communicates. Any Trigger type may be used to Trigger a Network Event.



A Trigger tells the Network Event WHEN to move the data within the Network Event.

In This Section

Master Protocol List	194
Slave Protocol List	196
Using Network Ports	198
Using a Dialup Network Port	202
Using a Cellular Modem Network Port	210
Using Network Sessions	213
Using Network Destinations	240
Modbus Protocol	248
Secure Data Exchange (SDX and STM) Protocols	287
DF1 Protocol	301
DNP 3 Protocol	309
Ethernet IP Protocol	331
Hart Master Protocol	340
Bricknet Protocol	347
Numeric and Alpha Numeric Pager Sessions	366
Email Protocol	366
Creating an FTP Client Interface	373
Using Remote Scaling	380

Master Protocol List

The protocols listed here are used as either master only or peer to peer and have the ability to initiate Network Events to a Network Destination for both reading and writing data.

Hardware supported depends on the controller configured. Some controllers have varied support for serial handshaking lines such as DTR, CD, RTS and CTS. Those ports with all lines will support a dialup modem for those protocols that will support a dialup link. Consult the controller's installation and technical reference guide for details on which ports will support these devices.

ScadaBuilder will show a button for Dialup, TCP/IP, and Cell Modem in the Network Port dialog for those ports that support those devices. The Hardware Supported column below will show whether an external modem is required for the particular protocol.

If a device supports serial then it will also support any radio or radio modem. Some radio devices require RTS as transmit signal (Push To Talk). Consult the controller's installation and technical reference guide for details on which ports will support an RTS signal.

USB is supported for Pinnacle and later controllers where specified.

Protocol	Description	Hardware Supported
Modbus RTU Master	Standard Modbus Master RTU protocol used for communicating to Modicon and other Modbus RTU compatible slaves.	Serial RS-485 Dialup USB serial. Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.
Modbus TCP/UDP Master	Standard Modbus Master TCP or UDP used for communicating to Modicon and other Modbus TCP/UDP compatible slave devices.	Ethernet Serial/Dialup PPP Serial/Dialup Slip. Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.
Modbus Ascii Master	Standard Modbus Ascii protocol used for communicating with Modbus Ascii slave devices.	Serial RS-485 Dialup. Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.
DF1 Master	Used for Allen Bradley Compact/Control Logix, SLC and PLC 5 serial equipped devices.	Serial RS-485 USB serial. Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.
Hart Master	Standard Hart Master protocol with support for communicating with Hart slaves. Can also act as a Hart Secondary Master.	Serial USB serial Hart Modem Required. Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.

Protocol	Description	Hardware Supported
SDX (Secure Data Exchange)	Used as standard transport between ICL Pinnacle and later controllers as well as Sprite, Sprite II, Ascent and other Sentry modules available from ICL.	Ethernet Serial RS-485 Dialup USB serial Serial/Dialup PPP Serial/Dialup Slip. Pinnacle and later only.
STM (SDX over Text Message)	Used as standard transport between ICL Pinnacle and later controllers as well as Sprite, Sprite II, Ascent and other Sentry modules available from ICL equipped with cellular modems.	Serial or USB Cell Modem Required. Pinnacle and later only.
DGH Master	Used to read and write I/O from DGH and Bristol Babcock I/O. D1000, D1700, D2000, D3000, D4000 etc... non-Modbus modules supported.	Serial RS-485 USB serial. Pinnacle and later,
Ethernet IP Master	Used for Allen Bradley Compact/Control Logix, SLC and PLC 5 Ethernet equipped devices.	Ethernet Pinnacle and later only.
DNP 3 Master	Used for communicating with DNP 3 compatible slaves.	Ethernet Serial RS-485 Dialup USB serial. Serial/Dialup PPP Serial/Dialup Slip.
Bricknet	May be used for controller to controller communications as well as Picobrick, Microbrick, Sprite and other Sentry modules.	Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.
DFA Open Link	Used for communicating with Dexter Fortson RTU's and devices.	Serial RS-485 USB serial. Pinnacle and later.

Slave Protocol List

The protocols listed here are used as either slave only or peer to peer. Only peer to peer protocols have the ability to initiate messages as well as receive them.

Hardware supported depends on the controller configured. Some controllers have varied support for serial handshaking lines such as DTR, CD, RTS and CTS. Those ports with all lines will support a dialup modem for those protocols that will support a dialup link. Consult the controller's installation and technical reference guide for details on which ports will support these devices.

ScadaBuilder will show a button for Dialup, TCP/IP, and Cell Modem in the Network Port dialog for those ports that support these devices. The Hardware Supported column below shows whether an external modem is required for the particular protocol.

If a device supports serial then it will also support any radio or radio modem. Some radio devices require RTS as transmit signal (Push To Talk). Consult the controller's installation and technical reference guide for details on which ports will support an RTS signal.

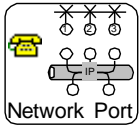
USB is supported for Pinnacle and later controllers where specified.

Protocol	Description	Hardware Supported
Modbus RTU Slave Network Message Links Supported.	Standard Modbus Master RTU protocol used for communicating to Modicon and other Modbus RTU compatible master	Serial RS-485 Dialup USB serial. Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.
Modbus TCP/UDP Slave Network Message Links Supported.	Standard Modbus Master TCP or UDP used for communicating to Modicon and other Modbus TCP/UDP compatible master devices.	Ethernet Serial/Dialup PPP Serial/Dialup Slip. Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.
Modbus Ascii Slave Network Message Links Supported.	Standard Modbus Ascii protocol used for communicating with Modbus Ascii master devices.	Serial RS-485 Dialup. Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.
DF1 Slave Network Message Links Required.	Used for Allen Bradley Compact/Control Logix, SLC and PLC 5 serial equipped devices.	Serial RS-485 USB serial. Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.

Protocol	Description	Hardware Supported
SDX (Secure Data Exchange) Peer to Peer Network Message Links Supported for Remote I/O master messages.	Used as standard transport between ICL Pinnacle and later controllers as well as Sprite, Sprite II, Ascent and other Sentry modules available from ICL.	Ethernet Serial RS-485 Dialup USB serial Serial/Dialup PPP Serial/Dialup Slip. Pinnacle and later only.
STM (SDX over Text Message) Peer to Peer Network Message Links Supported for Remote I/O master messages.	Used as standard transport between ICL Pinnacle and later controllers as well as Sprite, Sprite II, Ascent and other Sentry modules available from ICL equipped with cellular modems.	Serial or USB Cell Modem Required. Pinnacle and later only.
Ethernet IP Slave Network Message Links Required.	Used for Allen Bradley Compact/Control Logix, SLC and PLC 5 Ethernet equiped devices.	Ethernet Pinnacle and later only.
DNP 3 Slave Network Message Links Required.	Used for communicating with DNP 3 compatible slaves.	Ethernet Serial RS-485 Dialup USB serial Serial/Dialup PPP Serial/Dialup Slip. Pinnacle and later only.
Bricknet Peer to Peer Network Message Links Not Used.	May be used for controller to controller communications as well as Picobrick, Microbrick, Sprite and other Sentry modules.	Serial RS-485 Dialup USB serial Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.

All protocols in this list optionally use, or are required to use Network Message Links. See *Using Network Message Links* (on page 272) for more details. Each protocol will be slightly different depending on data types supported.

Using Network Ports



See *The ScadaBuilder Hierarchy* (on page 73)

To create a Network Port, double-click on “Communications” in the Project Manager or **Setup | Network Ports** menu. Each hardware port is listed as a tab. To configure a port, click on its tab.

Network Ports

Com1 Com2 (485) Radio Modem Com5 Usb1 Usb2

Device name:
Name:
Com2

Serial parameters:
Baud:
9600
Data bits:
8
Stop bits:
1
Parity:
none

Timing parameters:
Response delay:
0 (ms)

RTS Control:
☐ Always on
☐ Always off
☒ Transmit on
Transmit delays (ms):
0 Lead 0 Trail
RS-485 Default

Media Ready Mode:
☒ Always ready
☐ Time based
Receiver quiet time:
100 (ms)
Media access delay (ms):
0 Min 50 Max

Buffers:
Receive buffer size: Transmit buffer size:

OK
Reset
Help
Cellular
ICP/IP
NetSessions



Special Notes regarding the Ethernet Port The addressing and access controls for Ethernet are set up under the **Node | Settings | Ethernet / Serial IP** (see "Node Settings - Ethernet Tab" on page 37) tab in the Project Manager. The Ethernet port always shows up in compatible lists as "Ethernet".

Network Ports Reference

The Network Port defines and configures network communication ports. Network Ports can be referenced from Network Session to assign a communications protocol or attach a Textual User Interface to a port.

Name

Identifies the Network Port. While ports have default names such as Com1, Com4 or Modem, they may be changed to reflect their function. The name is how the Network Port is referenced throughout the rest of the ScadaBuilder application. Changing the name from the default is not strictly necessary but may be useful when configuring Network Sessions and other interfaces that use the port.

Baud

Specifies the serial data rate for the port. This must be set the same as any equipment you are attaching to in order for communications to succeed.

DataBits

Parameter specifies the number of data bits to use for the serial port interface. This must be set the same as any equipment you are attaching to in order for communications to succeed.

Stop Bits

Specifies the number of stop bits to use for the serial port interface. This must be set the same as any equipment you are attaching to in order for communications to succeed.

Parity

Specifies the parity setting to use for the serial port interface. This must be set the same as any equipment you are attaching to in order for communications to succeed.

RTS Control

Configures how the RTS (Request To Send) signal should be controlled. Refer to the hardware reference manual for your controller to find out which ports support the RTS signal. On RS-485 ports it is used to control the direction of information flow (RTS ON for transmit, OFF for receive). On RS-232 ports that support the signal, it is often used to key an external radio.

RTS Control settings:

- Always On The RTS signal is always in an on (active) state, regardless of whether any data is being transmitted.
- Always Off The RTS signal is always in an off (inactive) state, regardless of whether any data is being transmitted.
- Transmit On The RTS signal is on (active) whenever data is being transmitted and off when data is not being transmitted. In this mode, the Lead Delay and Trail Delay control the timing of the RTS signal. This mode is used for RS-485 and keyed-radio communications. On fast networks, the Lead Delay and Trail Delay may need some tuning to work reliably.

Lead Delay

Controls the timing of the RTS signal when the control mode is set to Transmit On. The Lead Delay is the time from when the RTS signal is asserted to when the data transmission starts.

Trail Delay

Controls the timing of the RTS signal when the control mode is set to Transmit On. The Trail Delay is the time from when the last byte of the data packet is transmitted to when the RTS signal is made inactive.

RS-485 Default

Click this button to automatically set the RTS control to the RS-485 defaults. This will set the mode to "Transmit on" and the lead and trail delays to 3ms and 1ms respectively.

Media Ready Mode

Applies only to the peer to peer SDX and BrickNet protocol over serial ports because of the potential for collisions.

The system works from reading bytes on the bus. When a byte is received the media is not ready according to the time parameters of the media ready configuration. The system holds off transmitting any message for the configured time plus a random time.

Care should be used when configuring this functionality. The greater the time parameters in the following fields, the slower the communication events will be in initiating data transfers. Large times can have a detrimental effect on polling round trip times (the time to service all destinations in the poll cycle).

Receiver Quiet Time

The amount of time that the receiver must be quiet for (no incoming data) before the media channel is considered ready (not busy).

Media Access Delay

Used only for BrickNet, when the Media Ready Mode is set to Time Based.

Once the media is determined to be ready, the Node waits a random period of time between Min and Max before checking the media again and transmitting a message. This is used to help avoid collisions when two nodes are waiting to transmit.

Response Delay

The time to wait before transmitting a response to a received message. This parameter is used to slow down responses from a slave or peer for networks that have poor message turnaround capabilities.

After receiving a valid message, the slave or peer will wait this period of time before responding to the message. This allows half duplex radios, for example, to settle back into receive mode so they are ready to receive the response.

Transmit And Receive Buffer Size

These settings allow you to override the default transmit and receive port buffer settings. It is recommended that you use this parameter only to increase the buffer sizes, and not to decrease them. The default buffer sizes depend on the protocol:

Protocol	Default Transmit Buffer Size (bytes)	Default Receive Buffer Size (bytes)
Modbus	256	256
DF1	512	512
BrickNet	512	512
DNP3	1024	1024
VUI	2048 (cannot modify)	64
TUI	2200	128
GPS	XXX	512
Dialer (Voice)	2048 (cannot modify)	64
Dialer (Pager)	64	64
Dialer (TAP)	64	64
Hart	64	64
Email (Dialup)	64	64
FTP (Dialup)	64	64

In general, these buffer levels are set automatically by the Network Session and other interfaces and do not need to be set when using a Network Port.

Reset Button

The Reset button resets all the Network Port parameters to their default/uninitialized states.

Dialup Button

The Dialup button takes you to the dialup modem parameters window for com ports that support a dialup modem.

TCP/IP Button

The TCP/IP button takes you to TCP/IP port parameters for configuring SLIP, CSLIP and PPP interfaces over a serial line. Generally, this is used in conjunction with a dialup modem.

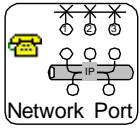
Cellular Button

This button takes you to the configuration of the cellular modem interface for this Network Port. Cellular modems are used for the Text Message Interface (TMI) and for configuring telemetry over STM (Secure Data Exchange over Text Messaging).

Network Sessions Button

This allows you to create and access Network Session(s) that are applied to the Network Port. It will also allow you to create a new Network Session on the fly.

Using a Dialup Network Port

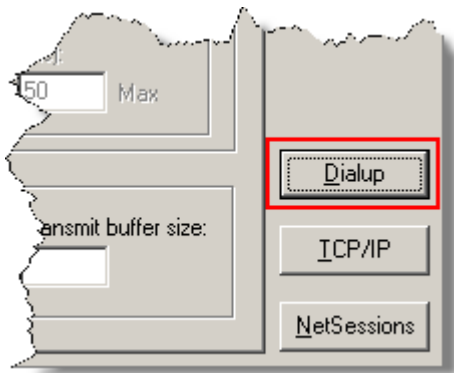


See *The ScadaBuilder Hierarchy* (on page 73)

Any controller serial port with the required control interface signals can be configured to support a modem; either internal or external. As a “slave”, or “server” the controller answers the telephone and responds to messages using the selected protocol. As a “master” or “client”, the controller will initiate a telephone call to a remote site each time a message needs to be sent.



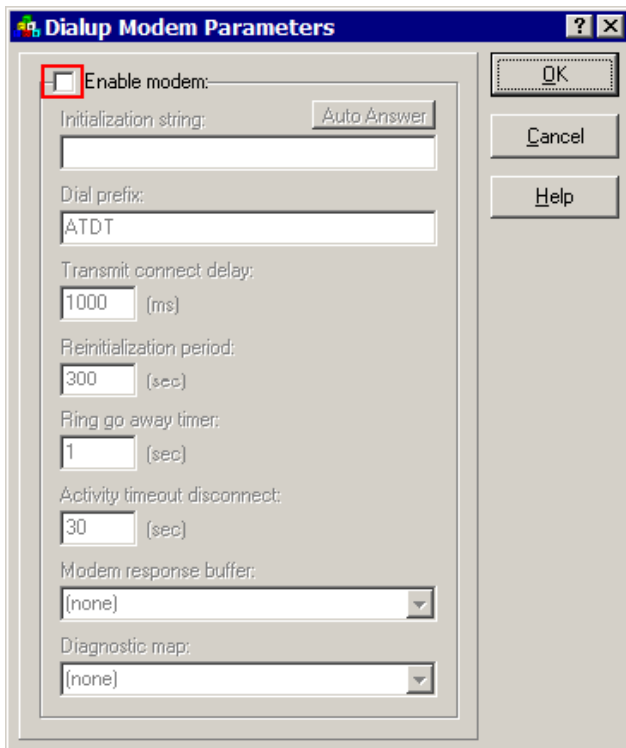
*If operating on an Internal Dialup Modem (the Modem option is selected in the **Node | Settings | General Tab | Controller Options**, (see “Controller Options” on page 35) then a small telephony icon will appear above the Dialup button. This means that your Modem is enabled.*



To set up a port for modem operation, click on the “Dialup” button in the lower right-hand corner of the Network Port window, bringing up a Dialup Options window.



If the dialup button does not appear, then the port does not support dial up feature on the controller configured. The port will be missing essential modem control signals like DTR, CD, CTS, and RTS.



In the Dialup dialog Check the “Enable Modem” box:

The initialization string is a configuration message that is sent to a modem to “train” it (auto baud). Most modems use “AT” commands for configuration. For example, the ATSO=1 command enables “auto-answer” and selects the number of rings (one) before the telephone line is picked up. Clicking on the “Auto Answer” button automatically inserts this command for you. You may have special requirements that can be added to the initialization string.

When calling out with voice or pager protocol (which are “connectionless”) the Ring Go Away Timer allows the controller to sense when a line is answered by sensing that “rings” are no longer detected for the specified period of time. It uses this timeout as a “connect” to the listener. In the United States, a value of 6 seconds is recommended.



Using the Ring go away timer with the voice Dialer feature causes pregnant pauses after a user picks up the phone. It sounds like an automated telemarketer dialing system is waiting to connect you to the next representative. This makes some folks instinctively hang up the phone after a few seconds.

Click OK to go back to the Network Port dialog.

Uses for Dialup Modems.

ICL Controllers can connect to a variety of dialup connections including:

- Controller to controller communication (Modbus or Bricknet). See the ***Modbus Protocol*** (on page 248) or ***Bricknet Protocol*** (on page 347) sections for more details.
- Internet Service Providers for Email, Modbus TCP/UDP, and FTP connections.
- Numeric and Alpha Numeric pagers.
- Dial In PPP server for remote debugging, viewing TUI's or transferring files off the controller.
- Dialer (voice) type alarming.
- Dial In Voice User Interface (VUI).
- Trending and User Portal Access.



To use any of ICL's voice functions (Dialers or Voice User Interface (VUI)), the ICL dialup modem option must be installed on the controllers.

Dialup Modem Parameters Reference

When a dialup modem is used on a Network Port, there are additional parameters to set.

Enable

ScadaBuilder will treat the communications port as a dialup modem port only if you enable it with the "Enable modem" checkbox.

If the port is an internal modem, the Enable modem check box will automatically be checked.

Activity Timeout

Any activity on this port will update this time including TCP/IP activity from FTP, HTTP or Telnet (TUI) interfaces. The Modem Activity Timeout and the TCP/IP Default Session Activity Timeout (if TCP/IP is used) are compared by the system and the greater of the two is used.

Auto Answer Button

Click the button to automatically insert the auto-answer string "ATS0=1" into the initialization string below.

Initialization String

This string is sent to the modem in order to configure it. The initialization string is sent on Node startup , periodically at the rate configured by the Reinit Period or any time the monitoring interface reestablishes control of the port (after a dial out procedure for example).

To set the modem to answer incoming data communications calls automatically, enable "auto-answer" with the following initialization string: ATS0=1

Unless you want the modem to auto-answer, you would normally leave the initialization string blank.

Dial Prefix

This parameter is used when the Node dials a telephone number. It is used for pager, data communications and voice dialing. The complete dial string is formed from the prefix and the appropriate telephone number.

Transmit Connect Delay

Once a modem carrier connection has been made, the Transmit Connect Delay must pass before any data messages are sent out the connected port.

Reinit Period

Determines how often the Node sends the Modem Initialization String to retrain and reconfigure the modem. This accounts for external modems, which can be powered down or reset independently of the Node.

Ring go away timer

Specifies the time duration that must elapse between rings during call origination before the modem assumes that the remote side has answered the phone (gone off-hook).

Under Voice Dialer operation, this parameter can also be used to determine a "connection". Most phone systems have a 5 second ring. For the system to "catch" a ring go away, this parameter should be set to 6 or greater. This has the

disadvantage of often causing a long pause after the ringing phone has been picked up and when the system begins to announce alarms.

Response Buffer

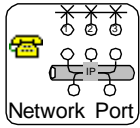
The modem response buffer can be used to select a buffer register (message) that holds response strings from the modem. This is primarily a debugging aid that is typically displayed on a TUI screen. Any commands sent to the modem such as AT and voice commands are echoed here. This is also a good place to see what phone numbers are currently being dialed by examining the "ATDT#####" commands sent.

Diagnostic Map Start Register

These diagnostic registers accumulate different modem functions and how often different events have happened when interacting with the modem and connections. Enter the start register (connect) as the first of the ten. The diagnostic values are always mapped in a contiguous block of 10 registers, as shown (in order):

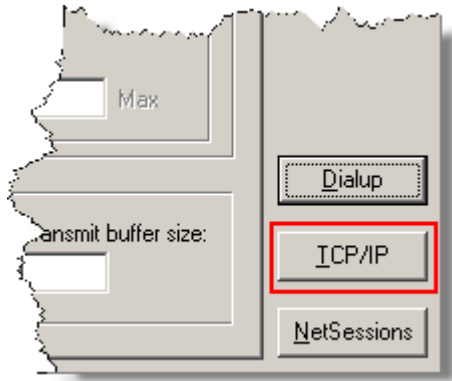
Connect	The number of successful connections made.
Ring	The number of rings seen by the modem.
No Carrier	The number of "no carrier" errors reported by the modem.
Error	The number of "error" responses from the modem.
No Dialtone	The number of "no dialtone" responses from the modem.
Busy	The number of "busy" responses from the modem.
No Answer	The number of "no answer" responses from the modem.
Response Unknown	The number of unrecognized modem responses.
Response Timeout	The number of modem response timeouts.
Connection Timeouts	The number of connection timeouts.

Using TCP/IP Over Serial and Dialup



See *The ScadaBuilder Hierarchy* (on page 73)

ICL controllers can be configured to run standard Ethernet protocols over serial links including modems and radios. Although you won't see the same performance as a "real" Ethernet connection, you can get the functionality benefits such as simultaneous file transfers HMI user interface access and security.



To set up a port for serial TCP/IP operation, click on the "TCP/IP" button in the lower right-hand corner of the Network Port window. This will bring up the "TCP/IP Port Parameters" dialog. If you have not enabled TCP/IP in the Node Settings, the TCP/IP button will not show in the Network Port Dialog. See *Node Settings - Ethernet / Serial IP Tab* (see "Node Settings - Ethernet Tab" on page 37) for more details.

You have a choice of three different protocols (means of authentication) to run serial TCP/IP. The choices come down to a trade-off between message overhead (important for slow links) and security.

SLIP - Serial Line Interface Protocol - is a protocol used to establish an IP network connection over an asynchronous serial link. There is no authentication.

CSLIP - Compressed Serial Line Interface Protocol - Similar to SLIP but faster and more compact. It reduces the TCP header from 40 bytes to 7 bytes.

PPP - Point to Point Protocol - Similar to SLIP, but supports password authentication, both CHAP and PAP, providing improved security.

If you don't have a preference, we generally recommend using PPP due to its wide acceptance and improved security.

In the "TCP/IP Port Parameters" window, selecting SLIP or CSLIP in the "Interface Class" field will gray out the remaining fields since this information is only required for PPP operation.



Each serial port on the controller may have its own "stack" meaning it can be (and in fact must be) a completely different network.

TCP/IP Port Parameters Reference

The TCP/IP Port Parameters dialog allows you to setup a TCP/IP interface on a serial line. The support interface classes are SLIP, CSLIP and PPP.

Whenever ISaGRAF is enabled and any of the TCP/IP parameters are changed in this dialog, the user is required to do a **Target | Send Complete Controller Setup...** from the main menu then restart the controller for the changes to take effect.

Enable Serial TCP/IP

This enables serial TCP/IP support for the current port. You can then select the desired interface class and optional services.

In addition, "Sockets" must be loaded onto the target. This can be accomplished through the ScadaBuilder Workbench by choosing **Target | Send Sockets...** menu when the desired Node is selected in the Project Manager window.

PPP Connect Timeout

This parameter allows configuration of the PPP connection timeout. Different ISP's (Internet Service Providers) as well as Cell Networks may take longer than the standard time to negotiate a PPP connection to allow TCP/IP and UDP communications. The system waits for the LCP, PAP and AP sequences (phases) to be completed before a connection is qualified as good. If the PPP Connect Timeout elapses before this happens, no connection is made and the system will hang up (disconnect) and try again from the beginning of the negotiation sequence. This usually involves dialing a Modem again.

Interface Class

The interface class specifies the specific protocol that is used to establish a TCP/IP serial interface. The supported classes are described below:

- SLIP - Serial Line Interface Protocol - is a protocol used to establish an IP network connection over an asynchronous serial link.
- CSLIP - Compressed Serial Line Interface Protocol - is a protocol like SLIP but faster on asynchronous serial links. It reduces the TCP header from 40 bytes to 7 bytes.
- PPP - Point to Point Protocol - is a protocol used to establish an IP network connection over serial lines and modems. It often replacing SLIP as it supports password authentication over dial up lines.

Enable Client Services

This enables client services when using the PPP interface class. This will allow the port to establish a connection with PPP server.

Use Authentication

This enables the password authentication protocol (PAP) when the PPP client services are enabled. The port will send the specified user ID password and password when trying to establish a connection with a PPP server.

User ID / Password

This specifies the user ID and password that will be used when the password authentication protocol (PAP) is enabled on a PPP class interface. The user ID and password are used by the PPP server to verify that the client is allowed to establish a connection. These are typically a user's account name and password when dialing into an Internet service provider (ISP).

Authentication Type

Authentication type depends on the Internet Service Provider or the host computer. In many cases PAP will work for an ISP but more "Nationwide" ISP's are configuring their systems for CHAP. Both PAP and CHAP require a username (UserID) and Password.

Enable Server Services

This enables server services when using the PPP interface class. This will allow other PPP clients to establish a connection with this PPP interface port.

Server IP Address

This specifies the local IP that will be used by the server/controller for the interface when a client logs in and establishes a PPP connection. This is the address the client or remote computer will be assigned to talk to the controller/server. This IP address should be of a different subnet than ANY interface on the computer logging in. Otherwise, problems will occur after the computer (client) connects.

Obtain DNS Servers

This option allows the system to configure itself to use DNS servers provided by a remote host such as a PPP dialup Internet Service Provider (ISP). This option also allows any Network Session to access a remote server by name rather than IP address.



This option can only be selected on one Network Port. Setting this option will turn it off on any other port that has it set.

This also selects the port as the default TCP/IP port of the controller. Any unspecified routes will attempt to go out this port from the TCP/IP stack. To avoid this problem, configure any routes desired over another port. To do this, go to **Node / Settings, Ethernet / Serial IP tab** (see "Node Settings - Ethernet Tab" on page 37) and click the Routing button. See **Node Settings - TCP/IP Routing Editor** (see "Routing Button" on page 49) for more details.

Client Login

The client login parameters consists of a User ID, Password and IP address (remote) that are used to establish a PPP connection with a server. The specified User ID and Password provide security that will be required by a remote client to login into the server services of the controller. The controller/server uses password authentication protocol (PAP) to validate the login. The specified IP address is that which will be assigned to the remote client after successfully logging into the server/controller.

Gateway (optional)

Allows the user to specify a remote gateway to allow routes through the host PPP server. This is especially useful if the host computer is sharing its Internet connection but does not provide gateway information automatically.

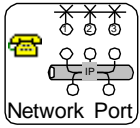
Default Session Activity Timeout

Any activity on this port will update this time including TCP/IP activity from FTP, HTTP or Telnet (TUI) interfaces.



The Modem Activity Timeout and the TCP/IP Default Session Activity Timeout (if TCP/IP is used) are compared by the system and the greater of the two is used.

Using a Cellular Modem Network Port



See *The ScadaBuilder Hierarchy* (on page 73)

Pinnacle and later controllers only.

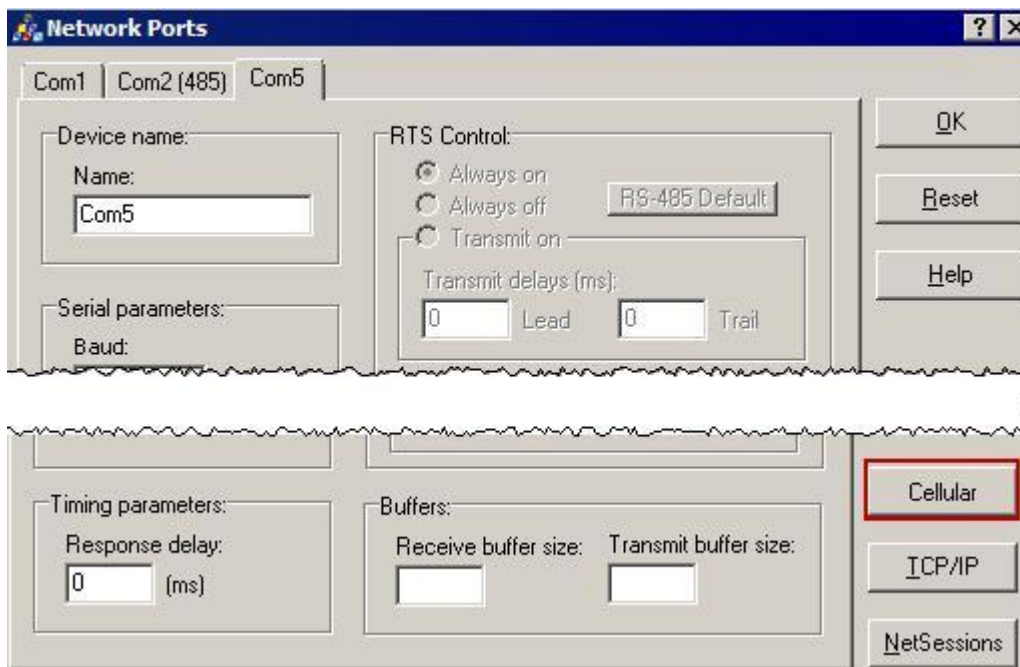
Any controller serial port with the required control interface signals or a USB port can be configured to support a cellular modem. As a “slave”, or “server” the controller receives the cellular message and responds to messages using the selected protocol. As a “master” or “client”, the controller will initiate a cellular message to a remote site each time a message needs to be sent. One caveat to using the interface is that the controllers Real Time Clock must be synchronized in some way. Old text messages are thrown out to prevent old messages from being delivered (the data may be completely out-of-date). See *RTC Sync Enable* (on page 212) and *RTC Sync Period (hrs)* (on page 212) in this section for details.

If operating a cellular modem on a port is possible then the **Cellular** button will show in the Network Port dialog. Once setup, the cellular modem can then be used for any available protocol or interface that can operate over cellular. The STM (SDX over Text Messaging) protocol can be used for reading and writing data to other controller and RTU's that support the STM protocol (and of course have a cell modem of their own. See *Setting Up An STM Interface* (on page 300) for details.

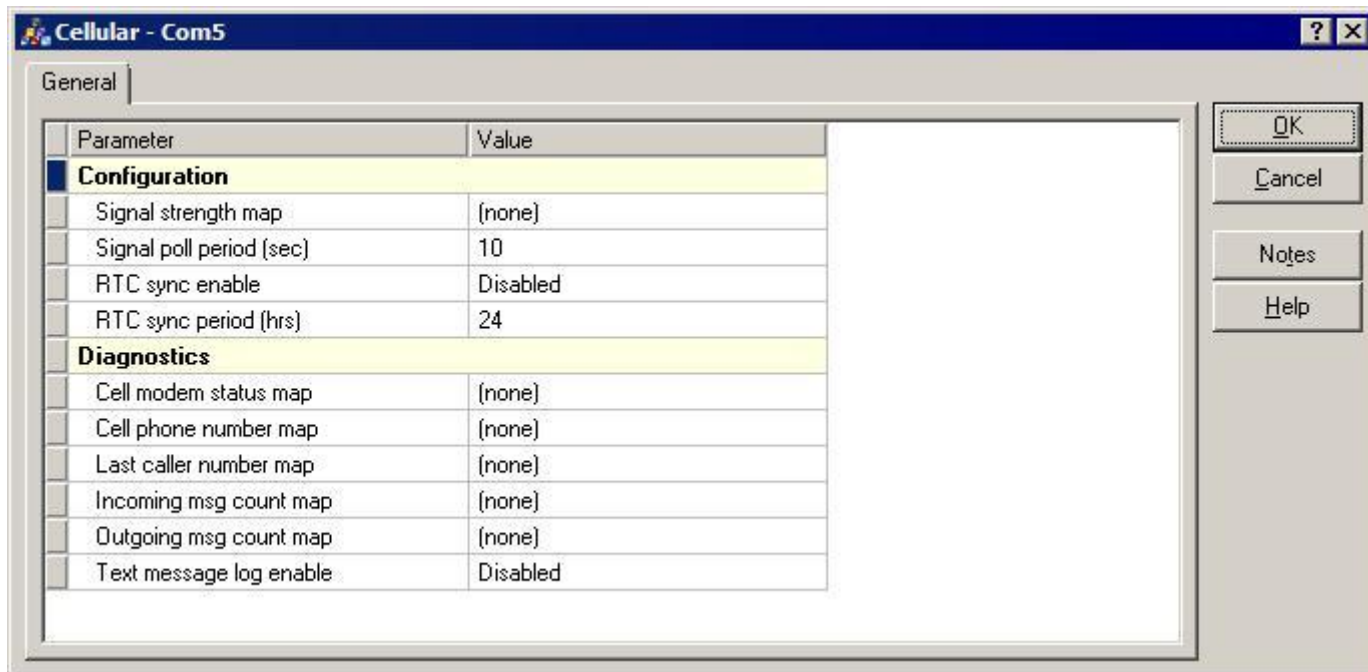
The TMI (Text Message user Interface) can be used with this port as well. See *Text Message Interface (TMI)* (on page 483) for details.

A Messenger series external cellular modem with a text message enabled account and SIM card is required to use this feature.

To configure it, click on the Cellular button in the Network Port dialog:



This will bring up the Cellular configuration dialog.



Nothing needs to be done with this interface by default but it does offer some useful services.

Signal Strength Map

Map an integer to get the signal strength from the currently connected cell tower.

Signal Poll Period (Sec)

How often to get data from the currently connected cellular tower including the current time for updating the Real Time Clock and Signal Strength.

RTC Sync Enable

When data is received from the cell tower, this flag will tell the system whether or not to synchronize the controller's internal Real Time Clock to that time. The Timezone and Daylight Savings parameters should be set or mapped to non-volatile registers set to the current TZ and DST for this feature to work properly. See **Time Zone Map/Constant** (on page 495) and **Daylight Savings Map** (on page 495) parameters in the **Mappings** (see "Mappings Reference" on page 493) section for more details.

RTC Sync Period (hrs)

How often in hours to synchronize the Real Time Clock to the data received from the currently connected cell tower. The Timezone and Daylight Savings parameters should be set or mapped to non-volatile registers set to the current TZ and DST for this feature to work properly. See **Time Zone Map/Constant** (on page 495) and **Daylight Savings Map** (on page 495) parameters in the **Mappings** (see "Mappings Reference" on page 493) section for more details.

Cell Modem Status Map

This status may be mapped to a Message register to view the current cell modem activity.

Cell Phone Number Map

Map a Message register to get the current phone number of the modem attached to the controller.

Last Caller Number Map

Map a Message register to show the last number that sent a message to this cell modem.

Incoming Message Count Map

Map an Integer register to count how many messages have been received by the local cell modem.

Outgoing Message Count Map

Map an Integer register to count the number of message sent out from this cell modem.

Text Message Log Enable

With this option turned on, each text message received and transmitted will be logged to CELL_DEBUG.log file located on the root of the IDE (c:) drive.

Using Network Sessions

Network Sessions represent the communication language the Network Port is going to speak. Protocol, timing, data handling, message routing (store and forward), and diagnostic information are all configured in the Network Session to tell the controller how to communicate.



See *The ScadaBuilder Hierarchy* (on page 73)

Much of the power of the Network Session is not that it can use a certain protocol, but how it handles and monitors poor communications and recovers from it. It is also important to point out that the programmer has a lot of power to configure how the Session does its job. While all the parameters and configurations may seem daunting at first, they represent a full feature set of professional communication tools.



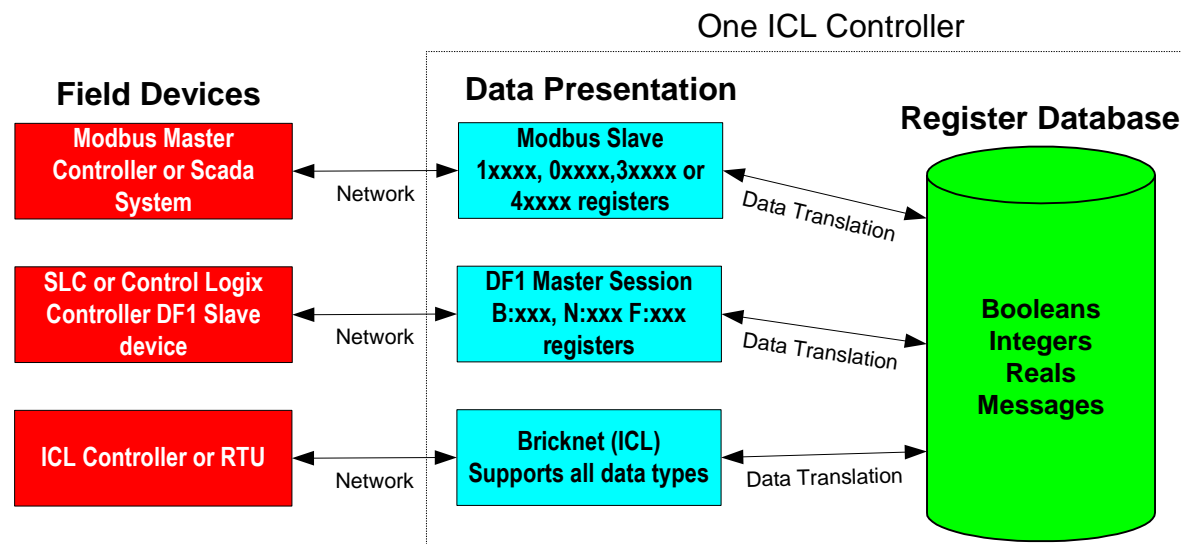
The Network Session tells the port HOW to talk.

See Communications Overview for more details.

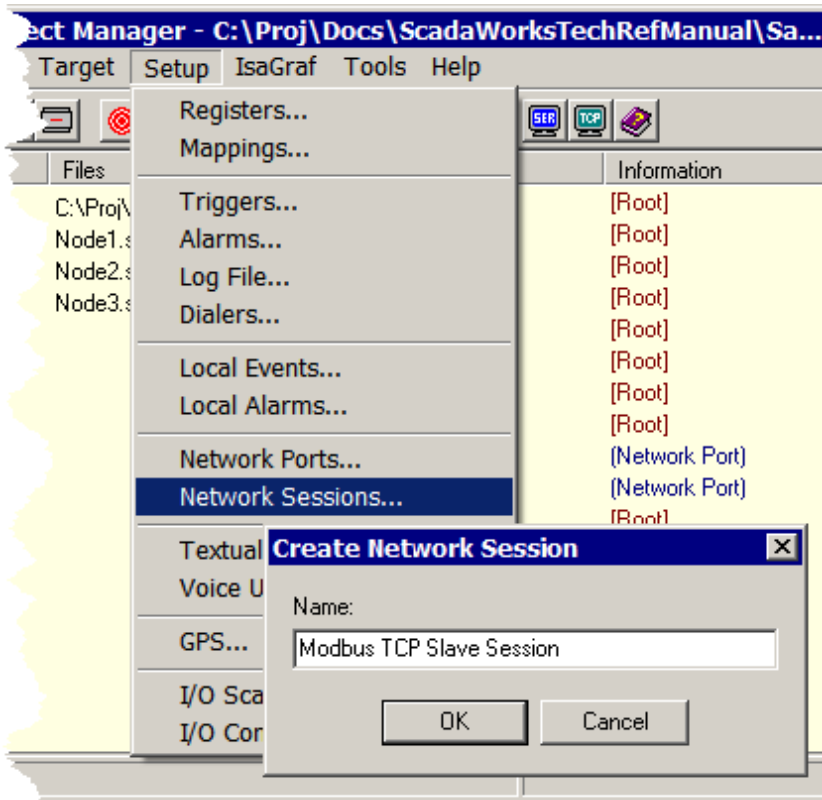
Data Presentation

One of the Network Session's jobs is to translate data to fit a particular protocol. For example, the Modbus numbering scheme and data type support does not natively handle floating point or long (32 bit) integers. The Network Session can translate floats (or Reals) into the natively supported 16 bit registers that Modbus expects. There is no standard for this process however and think of a way that a 4 byte (32-bit) number can be packed into 2 bytes (16-bit number) and index it, then someone has done it that way. The Network Session can handle all of these permutations.

This diagram shows that Network Sessions can also present the same controller data in many different ways:



Creating a Network session



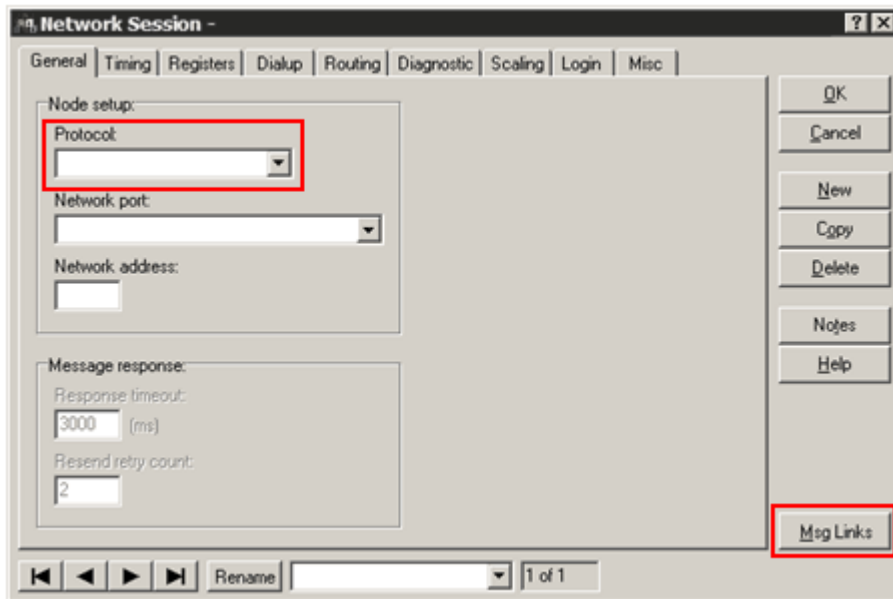
Once you have defined a Network Port, you must configure one or more Network Sessions to define how messages will be handled on that port.

There are several ways to create a Network Session: click on **Setup | Network Session** menu. Open a Network Port (in the Setup Tree, expand Communications and double click the Network Port you want, then click the Net Session button in the lower right hand corner) and a dialogue window will pop up for naming the Session.

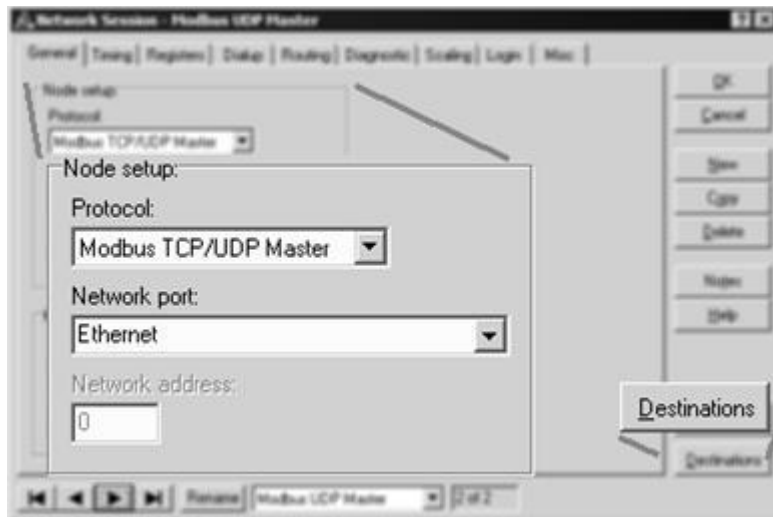
Accept the default name or enter a new one and click OK.

Once the Network Session is named, a Network Session configuration window is displayed.

Network Sessions have a great deal of functionality and configurability. Although there are nine configuration tabs for a Network Session, in reality, only the first of these eight tabs (General) is used for most simple applications. The other tabs support converting data formats, message routing, diagnostics and scaling of remote data. Although important, these features are used less frequently than the "General" configuration items.



Protocol Specific Configuration Buttons

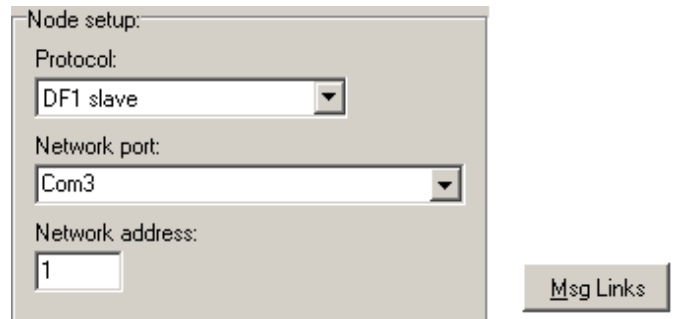


See *Network Events Reference* (on page 255) for more details.

In the the lower right-hand corner of the Network Session window is a button where the functionality "morphs" based on the protocol selected for the Network Session. The Protocol and function button are shown outlined in red above.

For any "master" or "peer-to-peer" protocol (such as ICL's SDX), there is a Destinations button. Clicking on this button leads to configuration windows to set up the Network Destinations and Network Events that initiate sending messages. See *Using Destinations* (see "Using Network Destinations" on page 240) more details.

For most slave and peer to peer type networks, a **Msg_Links** button handles special data configurations and for some protocols is a required configuration for data handling as there is no way to reconcile a default access list for all register data types. See the **Slave Protocol List** (on page 196) for more details on which protocols Network Message links are required and those that are optional. See **Using Network Message Links** (on page 272) and **Network Message Link Reference** (on page 274) for more information.



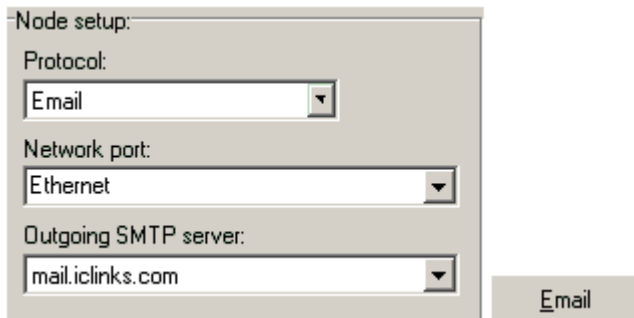
Node setup:

Protocol: DF1 slave

Network port: Com3

Network address: 1

Msg Links



Node setup:

Protocol: Email

Network port: Ethernet

Outgoing SMTP server: mail.iclinks.com

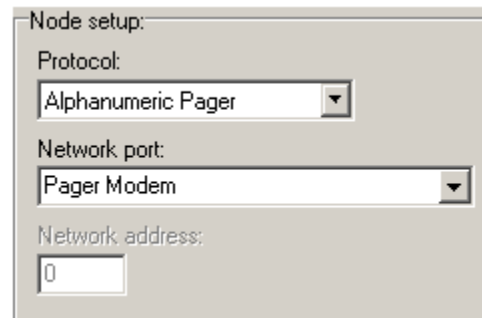
Email

An Email Network Session has its own button used to configure Emails. This includes the destination e-mail address, file attachments, the e-mail text including “live” register data and the “Triggers” that will initiate sending an Email. Emails can be sent over Ethernet or dialout modem connections. See **Creating an Email Interface** (on page 368) for more details.

Alarm annunciation protocols such as numeric and alphanumeric pagers have an **Alarms** button for configuring the Alarms that initiate the alarm annunciation and the data to be included in the alarm messages.



*These protocols are now configured in the Dialers setup. Paging protocols here are for backwards compatibility to old systems. See **Using Alarm Dialers** (on page 169) for more details.*

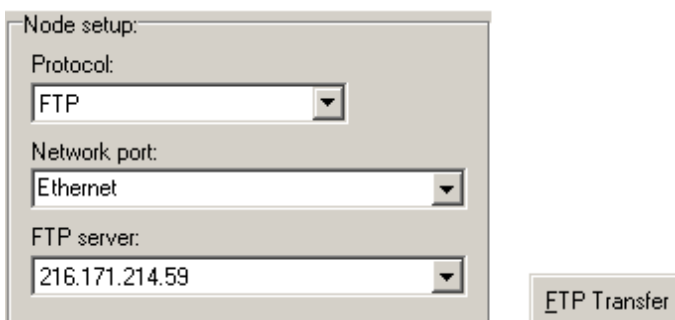


Node setup:

Protocol: Alphanumeric Pager

Network port: Pager Modem

Network address: 0



Node setup:

Protocol: FTP

Network port: Ethernet

FTP server: 216.171.214.59

FTP Transfer

When a Network Session is configured for the FTP protocol (FTP Client), an “FTP Transfer” button leads to the configuration window required to set up these transfers. This includes naming the files that will be transferred and defining the Triggers that will initiate the transfers. See **Creating an FTP Client Interface** (on page 373) for more details.

In Bricknet and Modbus Master (any type), if the File Transfer feature is enabled, you will see an additional button.



The File Transfer feature allows for the uploading and downloading of programs and log files over a serial style network such as Modbus RTU without affecting normal operation.

Node setup:

Protocol:
Modbus RTU master

Network port:
Modem

Network address:
0

File Transfer

Events

See **Creating Modbus Master Events** (on page 253) and/or **File Transferring Over Modbus and Bricknet** (on page 277) for more details.

As Network Sessions are created, the protocol which they use defines what fields are available in each dialog tab. For example, the Modbus RTU Master protocol shown above does not need a Network address parameter; there can only be one master (typically) for one network, therefore the Network address parameter is grayed out.

For each protocol, there are defaults in timing and retries that are optimized to work in almost any situation. Using the defaults is a good starting point and can get a system working quickly. Changing the defaults should be a matter of tuning the system for better throughput or more robustness. Parameters should not be changed haphazardly.



Research and understand a parameter before you change it. If you cannot get help from the documentation, call our technical support line at the front of this manual (see "Copyright Notice" on page 2) and we will be glad to help you.

Network Sessions Reference

The Network Sessions allow you to associate a protocol with a network port. For our purposes, numeric alarm paging (PAGER), alphanumeric alarm paging (TAP) and the Voice User Interface (VUI) are treated as protocols.

If multiple Network Sessions are defined for the same port, the last one defined determines the protocol that is used to monitor the port for incoming messages when no Network Events are being processed. This is important in certain situations, such as when VUI and TAP protocols are used on the same port (define the VUI Network Session last so that it is used to answer incoming calls).

This behavior can be overridden with the Monitor / Answer Default checkbox located on the "Misc" tab of the Network Session dialog.

A Network Session is applied to a Network Port. The properties of the Network Session's protocol determine what Network Ports are available. For example, if Modbus RTU Master is chosen as a protocol, only serial and dialup ports will show in the Network Port list box. If a TCP/IP protocol is chosen such as Modbus TCP/IP, Email or FTP, then only ports configured for TCP/IP will show.

Almost all of the connection timing, retries and communication status configuration is done in the Network Session. It essentially tells the system how to use a particular Network Port.

To tell the Network Session what data to send or how to treat received data, one must go one step further to define Network Events, File Transfer Events, FTP Events, Alarms, Emails or Net Msg Links to configure data transfers. These configurations are located in the lower right hand corner of the Network Session Dialog and are protocol dependent. See *Using Network Sessions* (on page 213) for more information.

General Tab

This tab contains configurations like protocol, Network Port, Slave address, timeout and retry parameters.

Protocol

Defines the protocol used for a Network Session.

The protocols listed here are used as either master only or peer to peer and have the ability to initiate Network Events to a Network Destination for both reading and writing data.

Hardware supported depends on the controller configured. Some controllers have varied support for serial handshaking lines such as DTR, CD, RTS and CTS. Those ports with all lines will support a dialup modem for those protocols that will support a dialup link. Consult the controller's installation and technical reference guide for details on which ports will support these devices.

ScadaBuilder will show a button for Dialup, TCP/IP, and Cell Modem in the Network Port dialog for those ports that support those devices. The Hardware Supported column below will show whether an external modem is required for the particular protocol.

If a device supports serial then it will also support any radio or radio modem. Some radio devices require RTS as transmit signal (Push To Talk). Consult the controller's installation and technical reference guide for details on which ports will support an RTS signal.

USB is supported for Pinnacle and later controllers where specified.

Protocol	Description	Hardware Supported
Modbus RTU Master	Standard Modbus Master RTU protocol used for communicating to Modicon and other Modbus RTU compatible slaves.	Serial RS-485 Dialup USB serial. Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.
Modbus TCP/UDP Master	Standard Modbus Master TCP or UDP used for communicating to Modicon and other Modbus TCP/UDP compatible slave devices.	Ethernet Serial/Dialup PPP Serial/Dialup Slip. Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.
Modbus Ascii Master	Standard Modbus Ascii protocol used for communicating with Modbus Ascii slave devices.	Serial RS-485 Dialup. Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.
DF1 Master	Used for Allen Bradley Compact/Control Logix, SLC and PLC 5 serial equipped devices.	Serial RS-485 USB serial. Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.
Hart Master	Standard Hart Master protocol with support for communicating with Hart slaves. Can also act as a Hart Secondary Master.	Serial USB serial Hart Modem Required. Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.
SDX (Secure Data Exchange)	Used as standard transport between ICL Pinnacle and later controllers as well as Sprite, Sprite II, Ascent and other Sentry modules available from ICL.	Ethernet Serial RS-485 Dialup USB serial Serial/Dialup PPP Serial/Dialup Slip. Pinnacle and later only.
STM (SDX over Text Message)	Used as standard transport between ICL Pinnacle and later controllers as well as Sprite, Sprite II, Ascent and other Sentry modules available from ICL equipped with cellular modems.	Serial or USB Cell Modem Required. Pinnacle and later only.

Protocol	Description	Hardware Supported
DGH Master	Used to read and write I/O from DGH and Bristol Babcock I/O. D1000, D1700, D2000, D3000, D4000 etc... non-Modbus modules supported.	Serial RS-485 USB serial. Pinnacle and later,
Ethernet IP Master	Used for Allen Bradley Compact/Control Logix, SLC and PLC 5 Ethernet equipped devices.	Ethernet Pinnacle and later only.
DNP 3 Master	Used for communicating with DNP 3 compatible slaves.	Ethernet Serial RS-485 Dialup USB serial. Serial/Dialup PPP Serial/Dialup Slip.
Bricknet	May be used for controller to controller communications as well as Picobrick, Microbrick, Sprite and other Sentry modules.	Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.
DFA Open Link	Used for communicating with Dexter Fortson RTU's and devices.	Serial RS-485 USB serial. Pinnacle and later.

The protocols listed here are used as either slave only or peer to peer. Only peer to peer protocols have the ability to initiate messages as well as receive them.

Hardware supported depends on the controller configured. Some controllers have varied support for serial handshaking lines such as DTR, CD, RTS and CTS. Those ports with all lines will support a dialup modem for those protocols that will support a dialup link. Consult the controller's installation and technical reference guide for details on which ports will support these devices.

ScadaBuilder will show a button for Dialup, TCP/IP, and Cell Modem in the Network Port dialog for those ports that support these devices. The Hardware Supported column below shows whether an external modem is required for the particular protocol.

If a device supports serial then it will also support any radio or radio modem. Some radio devices require RTS as transmit signal (Push To Talk). Consult the controller's installation and technical reference guide for details on which ports will support an RTS signal.

USB is supported for Pinnacle and later controllers where specified.

Protocol	Description	Hardware Supported
Modbus RTU Slave Network Message Links Supported.	Standard Modbus Master RTU protocol used for communicating to Modicon and other Modbus RTU compatible master	Serial RS-485 Dialup USB serial. Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.
Modbus TCP/UDP Slave Network Message Links Supported.	Standard Modbus Master TCP or UDP used for communicating to Modicon and other Modbus TCP/UDP compatible master devices.	Ethernet Serial/Dialup PPP Serial/Dialup Slip. Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.
Modbus Ascii Slave Network Message Links Supported.	Standard Modbus Ascii protocol used for communicating with Modbus Ascii master devices.	Serial RS-485 Dialup. Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.
DF1 Slave Network Message Links Required.	Used for Allen Bradley Compact/Control Logix, SLC and PLC 5 serial equipped devices.	Serial RS-485 USB serial. Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.
SDX (Secure Data Exchange) Peer to Peer Network Message Links Supported for Remote I/O master messages.	Used as standard transport between ICL Pinnacle and later controllers as well as Sprite, Sprite II, Ascent and other Sentry modules available from ICL.	Ethernet Serial RS-485 Dialup USB serial Serial/Dialup PPP Serial/Dialup Slip. Pinnacle and later only.
STM (SDX over Text Message) Peer to Peer Network Message Links Supported for Remote I/O master messages.	Used as standard transport between ICL Pinnacle and later controllers as well as Sprite, Sprite II, Ascent and other Sentry modules available from ICL equipped with cellular modems.	Serial or USB Cell Modem Required. Pinnacle and later only.
Ethernet IP Slave Network Message Links Required.	Used for Allen Bradley Compact/Control Logix, SLC and PLC 5 Ethernet equipped devices.	Ethernet Pinnacle and later only.

Protocol	Description	Hardware Supported
DNP 3 Slave Network Message Links Required.	Used for communicating with DNP 3 compatible slaves.	Ethernet Serial RS-485 Dialup USB serial Serial/Dialup PPP Serial/Dialup Slip. Pinnacle and later only.
Bricknet Peer to Peer Network Message Links Not Used.	May be used for controller to controller communications as well as Picobrick, Microbrick, Sprite and other Sentry modules.	Serial RS-485 Dialup USB serial Pinnacle and later, Etherlogic, ScadaFlex Plus and ICL4300.

All protocols in this list optionally use, or are required to use Network Message Links. See *Using Network Message Links* (on page 272) for more details. Each protocol will be slightly different depending on data types supported.

Network Port

Identifies which Network Port to use for this session. Only ports that can support the selected protocol will show in this list (TCP/IP ports for example).

Network Address

Used to identify this Node on the network. This is only used for the BrickNet protocol, Modbus slave protocols, DF1 slave protocol and DNP3 slave protocol.

The address can also be set via a register and on some ICL hardware, a switch. See the "Misc" tab, "Network address options" settings for the Network Session.

For Modbus TCP slave or Modbus UDP slave, the unit id must still be used.

Server

Used to identify the Server on the network that the session will connect to. This can be specified as one of three options--a numeric IP address, a host (server) name, or it can be retrieved from a message (buffer) register. This is used for Email and FTP protocols only.

Click on drop down box to select a buffer (message) register from which to retrieve a server name or IP address. An IP address can be specified as "xxx.xxx.xxx.xxx" in the buffer. If this address is not a valid 4 byte IP address then the system will try to look it up as a server name.

Host Name

To use the name of a server (i.e. mail.mydomain.com), you must configure the DNS (Domain Name Servers) parameter in the **Node / Settings / Ethernet / Serial IP tab** (see "Node Settings - Ethernet Tab" on page 37) to allow the controller to resolve the server name to an IP Address.

If using PPP dialup to an Internet Service Provider (ISP) you may use server names if you check the "Obtain DNS Servers" in the TCP/IP Port dialog under Network Ports | TCP/IP.

Response Timeout

The time to wait before resending a message to which there was no received response.

This is also used to specify the response timeout to VUI (Voice User Interface) prompts.

Under DNP3, response timeout applies to the "top" or application layer of the protocol versus the confirm and data link layers.

Resend Retry Count

The number of times to retry sending a message to which there was no received response before moving to the next Network Event or address in the Network Event List. When these retries are exhausted, the system will place that address in "Com Fail" and assert the Comfail flag if configured.

Under DNP3 protocol, this parameter applies to the "top" or application versus the confirmation and data link layers.

Confirm Timeout

If confirmation is required by the DNP3 protocol, this parameter controls how long the node will wait for a confirmation response. This parameter is not available for other protocols.

Confirm Retries

The number of retries the node will attempt before placing a DNP3 slave in communication failure. This parameter is not available for other protocols.

Data Link Timeout

Amount of time at the data link layer that the node will wait for an ACK or NACK from a remote DNP3 unit. This parameter is not available for other protocols.

Data Link Retries

Number of retries at the DNP3 data link layer before the node places the remote node in communications failure. This parameter is not available for other protocols.

Timings Tab

This tab contains most of the time dependent configurations for the Network Session.

Receive Character Timeout

Used only for slave/monitoring protocols: Bricknet, Modbus, DF1 and DNP3 slave. If a message is being received, and the incoming byte stream is interrupted for longer than this time, the message will be discarded and the system will wait for a new message.

Session Gap

Delay inserted after all pending event messages for a network session have been sent. Gap is used when more than one session share a Network Port.

Event Gap

Delay inserted after each individual event message in a Network Session is sent. This can be used to slow down the system polling cycle.

Probe Interval

Interval at which the Node will try to reestablish communications with a remote device that has failed to respond.



Communications failure is defined as the remote device not responding to any of the normal retries as set by the Response Timeout and Resend Retry Count parameters.

A value of 0 causes probing to occur at the rate configured by the Response Timeout. If an alternate Route has been specified, the Probe Interval must be set to a value greater than the Response Timeout. This allows Network Events to use the alternate Route if the primary fails (otherwise probes using the primary Route will dominate the port).

In general, the Probe Interval should be set to a value significantly greater than the Response Timeout to avoid tying up communications bandwidth trying to contact a node that is not responding.

Probe Interval Disable

The Probe Interval Disable check box allows the system to bypass the built-in communication failure recovery subsystem. When checked, it prevents the system from continuing to do retries after a comfail has been asserted and the system will wait for any fail address's Network Events to be retriggered. Using cyclic Network Events will nullify the effect.

Connection Timeout

Dialup

After dialing, carrier must be established within this time, or the Node will abort the connection.

TCP/IP

If a connection (socket) is not established with the remote unit within this time, the connection is aborted and the remote is considered to be in communications failure (probing will ensue).

Activity Timeout

This parameter applies to dialup and Ethernet only. If no messaging activity has occurred for this long, the session will be disconnected (modem hangs up for dialup, or socket disconnects for Ethernet).

Registers Tab

All parameter in this section apply to register based protocols. Data conversion, non-standard datatypes, truncation, and Modbus register access are all handled here.

AI Message Register Link

Required for Modbus slave protocols only. Identifies which register bank is associated with AI registers in Modbus messages.

Other names by which "AI registers" are sometimes known:

- Analog Inputs
- Input Registers
- 3xxxx registers

This parameter is not used in under ISaGRAF.

DI Message Register Link

Required for Modbus slave protocols only. Identifies which register bank is associated with DI registers in Modbus messages.

Other names by which "DI registers" are sometimes known:

- Digital Inputs
- Status Inputs
- Contacts
- 1xxxx registers

This parameter is not used in under ISaGRAF.

DO Message Register Link

Required for Modbus slave protocols only. Identifies which register bank is associated with DO registers in Modbus messages.

Other names by which "DO registers" are sometimes known:

- Digital Outputs
- Coils
- 0xxxx registers

This parameter is not used in under ISaGRAF.

AO Message Register Link

Required for Modbus slave protocols only. Identifies which register bank is associated with AO registers in Modbus messages.

Other names by which "AO registers" are sometimes known:

- Analog Outputs
- Holding Registers
- 4xxxx registers

This parameter is not used in under ISaGRAF.

Register Mode

Applies only to Modbus and DF1 protocols. Determines the byte ordering that is used to encode and decode registers in messages. (Both ends must agree on the same byte ordering).

MSB First*	The most significant bytes are sent first.
LSB First*	The least significant bytes are sent first.
InTouch*	Compatible with Wonderware® InTouch® software.
FIX*	Compatible with Intellution® FIX® software.
Auto	Applies only when ScadaBuilder is used in conjunction with ISaGRAF. Type conversions are automatically made between a 16-bit unsigned integer Modbus registers and 32-bit signed integer ISaGRAF variables. Does not support real (floating point) ISaGRAF variables.
Auto Signed	Applies only when ScadaBuilder is used in conjunction with ISaGRAF. Type conversions are automatically made between a 16-bit signed integer Modbus registers and 32-bit signed integer ISaGRAF variables. Does not support real (floating point) ISaGRAF variables.

This configuration may be overridden in Modbus Slave, DF1 Slave and DNP 3 Slave protocols for a register or block of registers by using the Network Message Link option. See Using Network Message Links section.

**To access a 32 bit number (Real or Integer) from a non-ICL device, take the ISaGRAF variable index and multiply by 2 and subtract 1 ($*2-1$) to access the proper register. The resultant index should always be an odd number.*

Integer Cast Type

Determines if integers in Modbus, Bricknet and DF1 messages should be treated as signed or unsigned values. This only applies when type conversions are taking place between the protocol message and target registers that are of a different type (int16 to int32 for example).

Index Mode

Applies only to Modbus protocols. Sets the mode used to determine the actual message start index for network events and network message links that use this session.

The Calculated mode bases the message start index on the transfer data type and message index specified for the event/link.

The Direct mode uses the message index specified for the event/link as the actual starting index.

Count Mode

This count mode is provided for compatibility with equipment that runs a variation of the Modbus protocol.

Standard

This mode is the most common. The register count in the Modbus messages refers to the number of 16-bit registers required to transfer the register data bytes. For example, if 5 floating-point registers (32-bits) are being transferred then the register count in the Modbus message will be 10. It takes 10 16-bit Modbus registers to transfer 5 32-bit floating-point registers.

Encoded

This less common mode interprets the register count in the Modbus messages as the number of registers of whatever data size that is being transferred (int16, float, etc.). For example, if 5 floating-point registers (32-bits) are being transferred then the register count in the Modbus message will be 5.

Dialup Tab

When a session uses a dialup interface for outgoing calls, all phone numbers, phone number lists and dialout retries are configured here.

If special modem Drivers are needed, those can be configured here as well.

The screenshot shows the 'Dialup' tab of a configuration window. At the top, there are several tabs: General, Timing, Registers, Dialup (selected), Routing, Diagnostic, Scaling, Login, and Misc. The main area is divided into three sections:

- Phone number list:** Contains a 'Number/Buffer:' label, a text input field with a dropdown arrow, an 'Add' button, and a list box containing the number '254-6732'.
- Phone number redial:** Contains a 'Redial wait (sec):' label with a text input field set to '30', and a 'Retry count:' label with a text input field set to '2'.
- Modem drivers:** Contains a 'Modem driver list:' label, a dropdown menu showing '(select driver)', a 'Load' button, an 'Unload' button, a 'Current modem driver:' label, and a text input field showing '(no driver loaded)'.

Phone Number

The telephone number specifies a specific number to add to the phone number list box when the associated "Add" button is clicked.

Phone Number Buffer

The name of a buffer that contains a telephone number to dial. Using a buffer instead of "hard coding" the telephone number allows you to change the telephone number while the application is running.

Redial Wait

Specifies the number of seconds to wait before retrying if the dialup connection fails.

Dial Retry Count

Specifies the number of times to retry if the dialup connection fails.

Current Modem Driver

This item displays the current modem driver configured for this Network Session. This allows the Network Session to be configured to use one of many modems. If this window shows (no driver loaded) then the default drivers are loaded for the standard ICL dialup modem option.

If a new modemdriver.ini file is install from an upgrade, the system will detect it and ask the user if they would like to update. Each modem driver has a name and a version number to reference it.

Modem Driver List

This dropdown box allows the selection of many "non-standard" modems. Cell modems, radio modems, V32 compatible drivers are supported from this list. To configure a modem driver, select the modem by name and click on the "Load" button. The modem's name should show in the "Current Modem Driver" box.

Load (Modem Driver)

Click here to load the currently selected modem driver in the Modem Driver List drop down box. This overrides the "standard" ICL dialup modem driver.

Unload (Current Modem Driver)

Click this button to unload the Current Modem Driver and return the Dialup Network Session to using the "standard" ICL dialup driver.

Routing Tab

All things to do with telemetry routing, address translation (Modbus), and Path (Bricknet) configuration.

Event Message Routes

Event Message Routes are used to do Modbus address translation on router nodes. They are only applicable to Modbus Slave Network Sessions. When a Modbus message comes into the Node, the address is examined to see if it is on the list of the "In Range" parameters in the routing list (these entries are seen after a route has been entered). If so, the address is then translated to the "Out Start" address offset of the qualifying routing entry and the message is retransmitted out the Network Port the Network Session is applied to. The remote will hear the resulting message and respond. The routing Node will again detect that the address needs to be translated again and repeats the procedure in the opposite direction.

Each routing entry contains the following parameters:

<i>In Start</i>	This parameter tells the Network Session what address(es) to look for and to do address translation on.
<i>Out Start</i>	This parameter is an offset to apply to any address in the In Start parameter.
<i>Block Size</i>	This parameter tells the routing node how many addresses to apply the address translation to.

The window will show the *In Range* and *Out Range* of each routing entry after it has been configured and the block size applied.

One caveat to using address translation is that no ranges or slave addresses can overlap. If there is a routing range from 31 to 40 in the "In Range" columns of ANY router in the system, then the addresses 31 to 40 cannot be used for remote slaves.

To delete an Event Message Route simply right click on the entry and select "Delete" on the speed menu.

In Start: (Address)

For Modbus routing only. Sets the start address to be used by routers in the system.

Modbus routing uses message address translation at the hops (or routing controllers) along the path which a message takes. The hop start address must be greater than the maximum slave address in the system.

Out Start: (Address)

For Modbus routing only. Sets the start address to be used for translation by routers in the system.

Modbus routing uses message address translation at the hops (or routing controllers) along the path which a message takes. The Out Start: address must be greater than the maximum slave address in the system.

Blocksize (Address)

For Modbus message routing only. This specifies the maximum address of any slave unit in the system. This information is used by the routing system to determine address translations.

Hop (Node Name)

Bricknet Only. Specifies which Node (along with the Bricknet Session) that the Network Event message should be routed through.

Network Session (Route)

Bricknet Only. On the hop node, specifies the Network Session that should be used to route the Network Event message.

Specifying this will require a **Target | Send Startup Config...** to that unit to make it functional.

New Path

Create a new Path to add Routers / Hops to.

Delete Path

Delete the current Path tab and with all its Hops.

Rename Path

Rename a Path tab.

Path Navigate

Use the << and >> buttons to navigate left or right through the available Paths.

Diagnostic Tab

All communication diagnostic statistics and information can be configured here.

Global Com Statistics

Network Session - STM Session

General | Timing | Registers | Dialup | Routing | **Diagnostic** | Scaling | Login | Misc

Com Stats | Com Fail | File Transfer

Global com statistic mappings:

Statistic: Register:

Statistic	Register
transmit command	(Integers 1001) Master_XmtCmd
receive response	(Integers 1002) Master_RcvRsp
receive command	(Integers 1003) Master_RcvCmd
transmit response	(Integers 1004) Master_XmtRsp
receive timeout	(Integers 1005) Master_RcvTmo
checksum error	(Integers 1006) Master_CrcErr
bad content	(Integers 1007) Master_BadCnt
configuration error	(Integers 1008) Master_CfgErr
success percent	(Integers 1009) Master_Prcnt
last rcv'd command	(Integers 1010) Master_LastMs

STM Session

Select the individual statistic you want to assign to an integer register or select (all) statistics and map them all at once.

Global Com Statistics

Specifies the communication statistic that is mapped to a register. Individual statistics can be mapped, or all available statistics can be mapped to one contiguous block of registers. The statistics are global in the sense that they apply to communications for all nodes on the network. Global communication statistics only apply to data protocols (like Modbus, Bricknet, DF1 and DNP3).

(all)

All available statistics values are mapped to the block of registers in the order listed below if it applies to the selected protocol.

Statistic	Explanation	Modbus / DF1		Hart	Bricknet	DNP3	
		Master	Slave			Master	Slave

Transmit Command	The number of command messages transmitted.	√		√	√	√	
Receive Response	The number of good response messages received.	√		√	√	√	
Receive Command	The number of good command messages received.		√		√		√
Receive Command	The number of response messages transmitted.		√		√		√
Receive Route	The number of messages to be routed that have been received.		√		√		√
Transmit Route	The number of routed messages that have been transmitted.		√		√		√
Receive Timeout	Applies to master or peer only (not slave). The number of times a message was sent and a response was expected but not received within the timeout period.	√		√	√	√	
Checksum Error	The number of messages received (addressed to this unit) which had a bad checksum or CRC error check code.	√	√	√	√		
Bad Content	The number of messages received which contained bad content (invalid content length, register type, register count, bad function code, etc.).	√	√	√	√	√	√
Configuration Error	Configuration error count. Indicates an invalid pager ID for TAP protocol (alphanumeric pager) or a data type mismatch for BrickNet protocol.				√	√	√
Lost Connection	This only applies to Modbus/TCP master. The number of times the TCP connection was lost unexpectedly.	TCP ONLY					

Success Percent	For master or peer only: the percentage of successfully completed transmit / receive message transactions. The percentage is calculated over the last 32 transactions and provides a rough indicator of the quality of the connection.	√		√	√	√	
Last Received Command	The time, in milliseconds since the last valid command message was received. This can be used to detect communication failure on a slave device.		√		√		√
Current Path ID	Bricknet only. Displays the current network routing path being used by number. As communication fails on one path, alternate paths may be used to complete data transaction's.				√		
DNP3 Statistics						Master	Slave
Confirm Timeout						√	√
Confirm Receive						√	√
Confirm Transmit						√	√
Local Error code						√	√
Remote Error code						√	
Indicator Bits						√	
DNP3 Link Layer Statistics							
Transmit Primary						√	√
Receive Primary						√	√
Transmit Secondary						√	√
Receive Secondary						√	√
Confirm Timeout						√	√
CRC Error						√	√
Fragment Overflow						√	√
Link Reset						√	√

Resp ACK						√	√
Resp NAK						√	√
Resp Busy						√	√
Resp Ready						√	√
Resp Service Error						√	√

Global Com Statistics Add Button

This button adds a statistic/register mapping to the list.

Global Com Statistics List

Lists the global communication statistics that have been mapped to registers.

File Transfer

The file transfer feature is supported for any Modbus or Bricknet protocol. Enabling the feature is done in the **"Misc"** tab (see "Miscellaneous Tab" on page 238) by clicking the "Enable File Transferring" checkbox. From there, simply click on the File Transfer button where you wish to initiate a file transfer.

The screenshot shows the 'Network Session - Node 1 Bricknet Session' dialog box with the 'File Transfer' tab selected. The 'File transfer mapping' section contains the following fields and controls:

- Transfer state register:** A dropdown menu.
- Current file bytes register:** A dropdown menu.
- Total file bytes register:** A dropdown menu.
- Current xfer bytes register:** A dropdown menu.
- Total xfer bytes register:** A dropdown menu.
- Transfer abort register:** A dropdown menu.
- Transfer abort (unlock) timeout:** A text box containing '60' followed by '(sec)'.
- Open file buffer:** A dropdown menu with '(none)' selected.
- Transfer packet size:** A text box containing '128'.

Buttons on the right side of the dialog include: OK, Cancel, New, Copy, Delete, Notes, Help, File Transfer, and Events. At the bottom, there are navigation buttons (back, forward, etc.), a 'Rename' button, and a session name dropdown set to 'Node 1 Bricknet Session' with a page indicator '1 of 1'.

Only "master" protocols may support file transfer initiation. Any Modbus Slave that you wish to transfer from must have the "Enable File Transferring" checkbox checked in the Modbus Slave Network Session.

The feature may be used for retrieving log files, updating applications or retrieving system.log files for troubleshooting.



File transfers over Modbus and Bricknet are proprietary to ICL Controllers.

Open File Buffer

The buffer (message) register will display the current file the transfer system is working with.

State Register

Maps the File Transfer State to a register. Only Integers may be used. The possible values of the Transfer State Register are as follows:

0 = Idle	There is no file transfer activity
1 = File Transfer Locked	Unit has been locked during which no other file transfers may happen in the system on this network.
2 = Initializing File Write	Currently in the process of looking for sufficient remote file space and validating the file name.
3 = Initializing a File Read	Currently in the process of looking for sufficient local file space and validating file name.
4 = Opening File in Question	The file is being opened in preparation for data transfer. In reality, the system operates on temporary files and renames them after the transfer is complete.
5 = Writing File Data To Remote	The file is being transferred packet by packet from the initiating Node to the remote Node.
6 = Reading File Data From Remote	The file is being transferred packet by packet to the remote Node from the initiating Node.
7 = File Close	Temporary file is being closed. There is no more data to transfer.
8 = File Clean Up	This stage is where the temporary file is copied to the destination name (local or remote). This allows the transfer mechanism to safely transfer system and application files as well as data log files.
9 = File Unlock	Allow other file transfers to happen in the system or continue on to the next file transfer operation.
10 = File Done	There are no more transfers to complete. The operation was successful.

Current File Bytes Register

Integer - this register shows the number of bytes transferred for the current file (shown in the Open File Buffer register).

This parameter may be used to calculate the transfer percentage of the current file. The calculation would be:

$(\text{Current File Bytes}) / (\text{Total File Bytes}) * 100$

Total File Bytes

Integer - Shows the total number of bytes being transferred for the current file (the current file size).

This parameter may be used to calculate the transfer percentage of the current file. The calculation would be:

$(\text{Current File Bytes}) / (\text{Total File Bytes}) * 100$

Current Xfer Bytes Register

Integer - This register shows the current number of bytes transferred in the current File Transfer Event. The File Transfer Event may span multiple files and could take a considerable amount of time. This parameter allows the user to monitor the progress of the transfer. The number may be used to calculate the percentage progress of the current File Transfer Event using the following formula:

$(\text{Current Xfer Bytes}) / (\text{Total Xfer Bytes}) * 100$

Total Xfer Bytes

Integer - This parameter shows the total number of bytes in a given File Transfer Event. This total includes all files in the Transfer Event. This total may be used in a transfer percentage calculation with the following formula:

$(\text{Current Xfer Bytes}) / (\text{Total Xfer Bytes}) * 100$

Abort Register

Boolean - setting this register to true during a transfer operation will abort the entire process.

Abort Timeout

Integer - This register defines the number of seconds to wait before aborting a transfer operation. Since all other transfer events are locked out during an active transfer, should a node stop responding, the initiating node will wait this time before unlocking itself. In the case of Bricknet, any node can then use the network for a file transfer at that time.

Packet Size

Integer - defines the number of bytes per message transaction to use for file transfer operation. This is a way to "throttle down" the message sizes. This is particularly handy for use with modems with small packetizing buffers that will not allow larger messages through. Maximum size for Modbus: 255. Maximum size for Bricknet: 512.

Create Host File Transfers

This function is used in conjunction with remote file hosting (see **Remote Host File Transfers** (on page 281)). This button will set up automatically the events necessary to carry out a remote file host update according to the Remote File Hosting configuration in the Node Settings dialog when multiple nodes are using Modbus or Bricknet across a multi-drop network (such as half duplex radio or RS-485.)

File transfers over Modbus and Bricknet are proprietary to ICL Controllers.

Scaling Tab

Remote Scaling of communication based register data can be configured here and applied to both incoming and outgoing data.

This feature can be used for scaling analog inputs and outputs being passed over the communication link through this Network Session. This is particularly useful for scaling AI's and AO's of ICL Microbricks, Picobricks, MAXIOs, and ScadaBridges.

Remote Scaling

Specifies the I/O Scaling entry to apply to the associated register. The scaling information in the selected entry will be applied to the register whenever it is read or written during communications in this session. To create a Remote Scale, go to I/O Scaling and select "Remote" as the I/O mode in the Scaling record.

Remote Scaling List

Lists the registers that have remote I/O Scaling entries applied to them.

Remote Scaling Register

Specifies the register that the I/O Scaling entry is applied to.

Login Tab

This tab is for FTP file transfers to a server as well as Secure Authentication for email (when it is enabled). Login information for the destination FTP server or SMTP email server is configured here.

Login User Name

This specifies the user name when logging into an FTP server or an Email Server that requires authentication.

You may select a string in the drop down tree and enter the data manual, select an existing Message (buffer), or create a new Message to hold the user name.

Login Password

This specifies the password when logging into an FTP server.

You may select a string in the drop down tree and enter the data manual, select an existing Message (buffer), or create a new Message to hold the password.

Use Login Authentication

Check this box to turn on Secure Password Authentication for those SMTP (email) servers that require it. ISP's (Internet Service Providers) use this to prevent their servers from being used for spam emailing. Older servers and local SMTP servers do not usually require authentication.

Use Active Authentication to Establish Data Link

Many newer Microsoft servers use active mode authentication while older, UNIX based systems tend to use passive mode (the ICL default). Check this box if you are having trouble authenticating to a Microsoft FTP server (IIS).

Miscellaneous Tab

The "Misc" contains parameters that are not necessary for everyday use of the Network Session but can provide a bit of extra functionality.

Network Session - Node 1 Bricknet Session

General | Timing | Registers | Dialup | Routing | Diagnostic | Scaling | Login | **Misc**

Session options:

- ☐ Monitor/answer default
- ☐ Disable session at startup
- ☐ Slave accept broadcasts
- ☐ Use BCC instead of CRC
- ☒ Enable file transferring

Status buffer:

Address reference:

Diagnostic status cmd map start (24 regs):

Alarm annunciation:

Alarm wait (min): Retry count:

List wait (min): Retry count:

TCP/IP:

Client limit: TCP port:

Account/Domain name:

☐ Use UDP instead of TCP

Navigation buttons: [Back] [Previous] [Next] [Forward] Rename: Node 1 Bricknet Session 1 of 1

Alarm Wait

Specifies the number of minutes to wait before re-annunciating an alarm via dialup (pager or voice).

Alarm List Wait

The time to wait before restarting the alarm phone number list.

Alarm Retry Count

Specifies the maximum number of times to re-annunciate an alarm that has not yet been acknowledged.

Alarm List Retry Count

The maximum number of times to restart the alarm phone number list.

Default Monitor-Answer Session

Setting this checkbox identifies the Network Session as the default session that will run when no other Network Session is active. This allows it to monitor for incoming messages and dialup connections.

This parameter also applies to any VUI Network Session on a modem port.

Disable Session at Startup

If this setting is checked, the Network Session will be disabled when the application starts. This prevents any message traffic on the Session until it is enabled. The application may enable the session using the NSCtrl function in your ISaGRAF application.

For normal applications, do not set this option.

Accept Broadcasts

Enables/disables incoming broadcast message detection. There is a slight performance improvement with broadcast disabled. Applies to serial Modbus Slave protocols only.

Use BCC Instead of CRC

Setting this checkbox causes the network session to use a BCC (block check character) field instead of a CRC (cyclic redundancy check) field at the end of each message. This control only applies to DF1 master and slave protocols. DF1 is always half duplex on ICL controllers and RTU's.

Protocol Status Buffer

Identifies a buffer used to show protocol status for debugging/monitoring purposes. Currently reports modem dialing and TCP/IP status information.

Network Address Reference

Determines where the Network Session address setting comes from:

Session	from the address setting on the "General" tab of the Network Session setup window.
Switch	from a hardware switch. This is supported only on selected ICL hardware. (If the switch is set to "0" then the address in the "General" tab of the Network Session setup window is used.)
Register	from the value stored in a register. (If the register value is set to "0" then the address in the "General" tab of the Network Session setup window is used.)

Network Address Register

The register from which the network address of the Network Session should be read. This allows the address to be easily reconfigured. This should always be mapped to a non-volatile (retained) register.

TCP/IP Client Limit

Applies only to Modbus/TCP slave protocol. Sets the maximum number of client connections that are allowed. This number must be less or equal to the *Node / Settings, Ethernet / Serial IP* (see "Node Settings - Ethernet Tab" on page 37) "TCP Max Sockets" parameter for the maximum affect.

Account/Domain Name

This specifies the account or domain name that is used in the "From" field when sending emails. If a full account name is specified (such as "myaccount@mydomain.com") then the text will be inserted into the "From" field as is.

If just a domain is specified (such as "mydomain.com") then the node name will be pre-pended to the domain to create an account name. For example, if the node name is "Node1" and the domain is "mydomain.com" then the text that will be inserted into the "From" field will be "Node1@mydomain.com".

TCP Port

Specifies the TCP port that the session is to use when establishing a TCP/IP connection. Applies to Modbus/TCP master and slave protocols as well as Email and FTP Transfers.

Use UDP Instead of TCP

This option converts a Modbus TCP Master or Slave session into a UDP Master or Slave. While TCP is a connection based protocol, UDP works more like a serial link--there is no connection to establish or maintain. UDP is ideal for physical media that might drop out unexpectedly (like a cell or radio connection).

Enable File Transferring

For Modbus Master and Slave protocols as well as Bricknet, enabling file transfers allows the user to transfer logs or new programs "over the air." on a serial network.

Using Network Destinations

In setting up data communications, Network Destinations are the third stage to configuring a successful data path to another controller or RTU device.

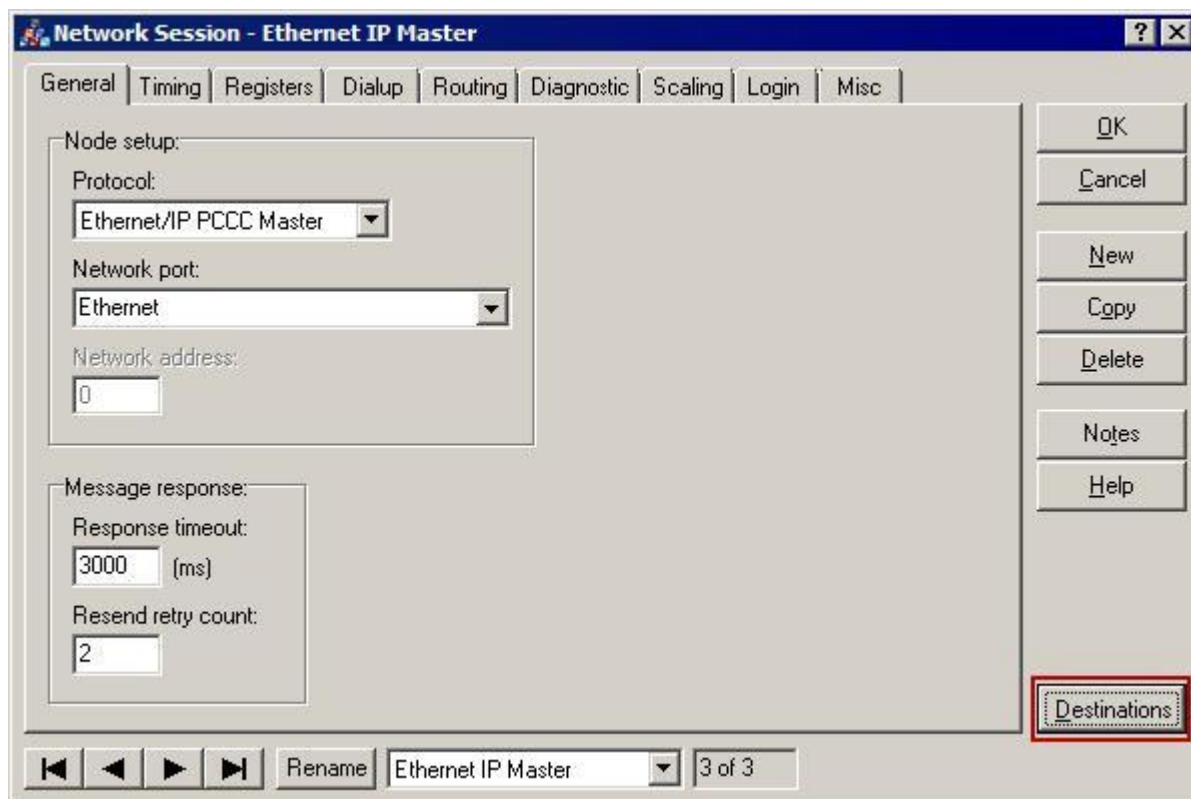


See *The ScadaBuilder Hierarchy* (on page 73).

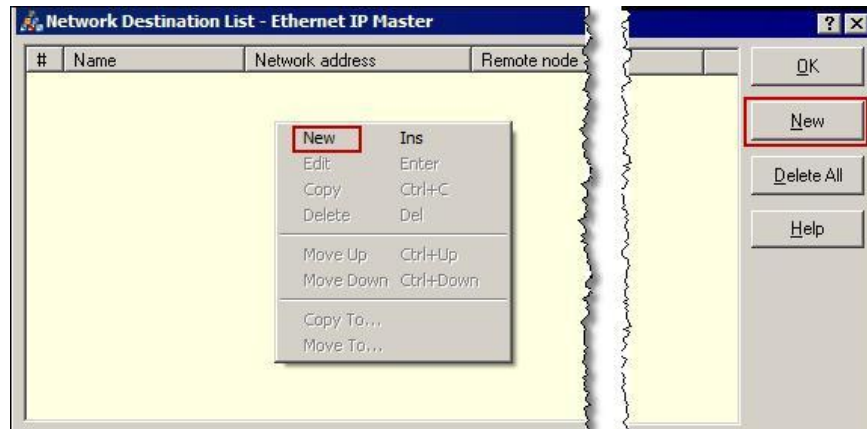
A Network Port must first be setup for proper communications to the hardware of a port. See *Using Network Ports* (on page 198) for more details. Also for reference See *Using a Dialup Network Port* (on page 202) or *Using a Cellular Modem Network Port* (on page 210) depending on your protocol's needs and compatibility. See *Master Protocol List* (on page 194) for protocol / Network Port compatibility.

A Network Session is then used to create the protocol and tell the system how the Network Port will be used. Each protocol is a little different and will change how subsequent records like Network Destinations, Network Event and Triggers are used. See *Using Network Sessions* (on page 213) and the particulars of each protocol is covered in similar section in this manual. All the configurations are too numerous to go into here but the particular protocol will change the way data is used and configured depending on that protocol's capability and supported features.

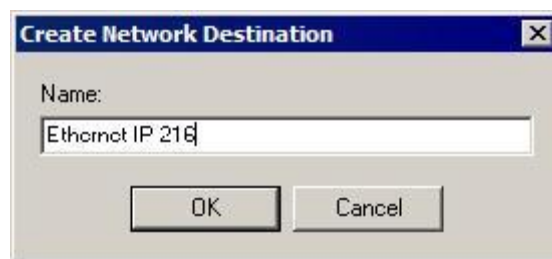
Create the Network Port and Network Session. In the master or peer to peer protocol you will see a Destinations button in the lower right hand corner that will take you to the Network Session's Network Destination list:



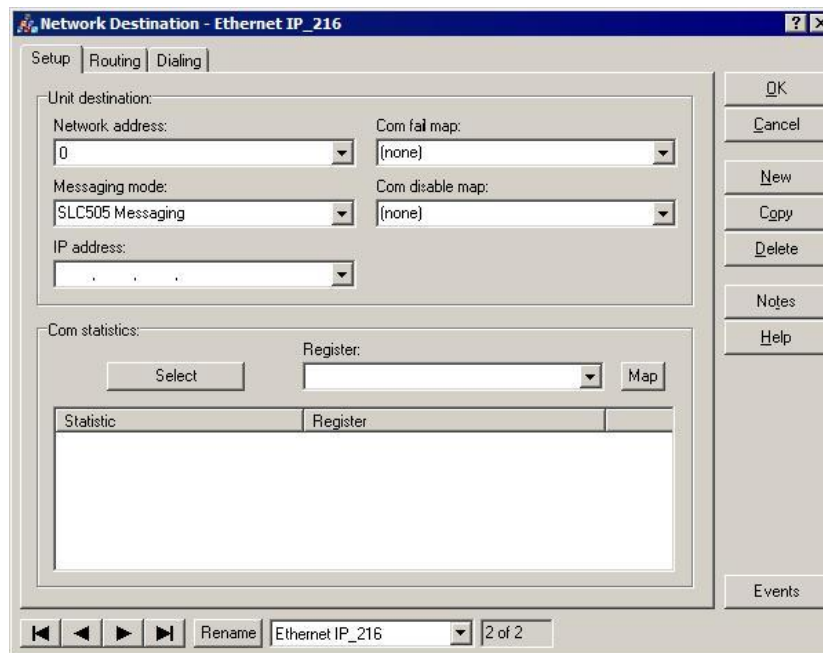
From the Network Destination list you can click the New button or right click and select New...



You will be prompted for a Network Destination Name. You can call it anything you like but it is suggested that you make it a meaningful name that is easily recognizable in the ScadaBuilder Setup Tree.



Depending on the protocol the required parameters change. In this example we are talking with an Allen Bradley over Ethernet IP so the Node Address and IP address are required:



So we fill in the required parameters but we also have the ability to add appropriate statistics, a communication failure flag for this device and a communication disable flag to turn off all events to this particular destination.

Network Destination - Ethernet IP 216

Unit destination:

Network address: 1

Com fail map: CommFail_216 (10001)

Messaging mode: SLC505 Messaging

Com disable map: ComDisable_216 (10002)

IP address: 192.168.1.216

Com statistics:

Statistic	Register
transmit command	(Integers 10003) XmtCmd_216
receive response	(Integers 10004) RcvRsp_216
receive timeout	(Integers 10005) RcvTmo_216
bad content	(Integers 10006) BadCnt_216
configuration error	(Integers 10007) CfgErr_216
lost connection	(Integers 10008) LstCnt_216
success percent	(Integers 10009) Percnt_216

Buttons: OK, Cancel, New, Copy, Delete, Notes, Help

Navigation: [Previous], [Next], [First], [Last], [Rename: Ethernet IP 216], [1 of 1]

Since the protocol supports several device types, we must choose the Messaging Mode parameter as well. These options are protocol dependant. Each protocol might be a little different. For example, an IP address will not show up when configuring a Modbus RTU (serial) Destination but the Network Address is still required. If the parameter is required then ScadaBuilder will not let you exit the dialog until all required parameters are done or the Cancel button is clicked. In the case above, neither Routing or Phone dialing are required so the fields in those dialogs are disabled. Once a Network Destination is created and our remote unit's parameters are defined, then we can click the (Network) Events button to configure what data we are sending.

Click OK here when done with the Destination click on Events to configure what data needs to be moved in and out of the remote unit's registers or I/O.

To create another Network Destination, click on the New button.

You can create as many Network Destinations as you have remote units limited only by the protocol's addressing limitations. For example, a Modbus Network Destination may have only 253 addresses total (1-254).

Network Session - EthernetIP

General | Timing | Registers | Dialup | Routing | Diagnostic | Scaling | Login | Misc

Node setup:

Protocol:

Network Destination List - EthernetIP

#	Name	Network address	Remote node	IP Address
1	CompactLogixL23E	0		192.168.237.217
2	MicroLogix1400	1		192.168.237.218
3	SLC_505	1		192.168.237.219

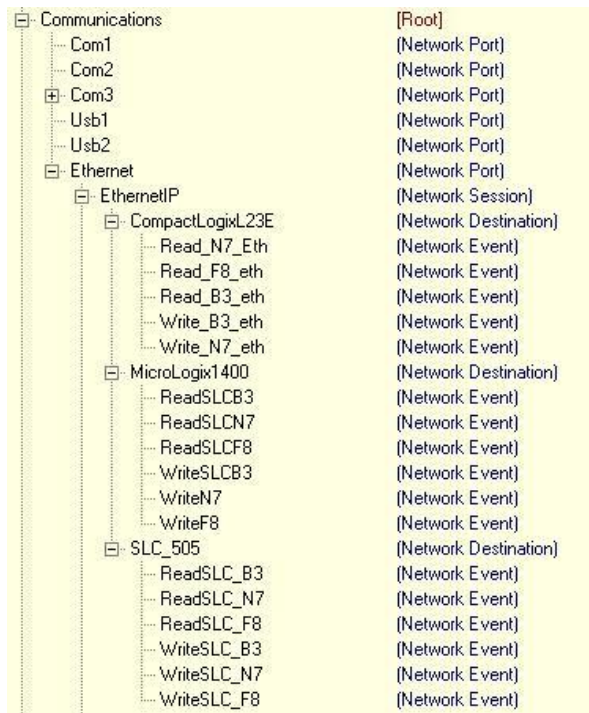
Buttons: OK, Cancel, New, Delete All, Help

Resend retry count: 2

Destinations

Navigation: [Previous], [Next], [First], [Last], [Rename: EthernetIP], [3 of 4]

When a Master or Peer to Peer Network Session services the Network Destinations, they are handled in the order they are defined. Network Events under and Network Destination are handled in the order they appear in the Network Destination's list. Only Network Events that have been Triggered or are pending based on time will actually send a message. Otherwise, the Network Session will move on to the next Network Event/Destination combination in the Network Session's Network Destination list.



You do not need to worry about Network Destination and Network Event combinations stepping on one another. The communications engine will complete each transaction (or timeout) and move on to the next in an orderly fashion. If there are multiple Network Sessions on a serial port, they will also be handled in the order configured. If there are multiple Network Sessions on an Ethernet port, then each session will run independently.

Network Destination Reference

Setup Tab

Statistic	Register
transmit command	(Integers 1) xmtcmd
receive response	(Integers 2) rcvrsp
success percent	(Integers 3) percent

Address

Used to identify the remote node on the network that the message is destined for.

Remote Node

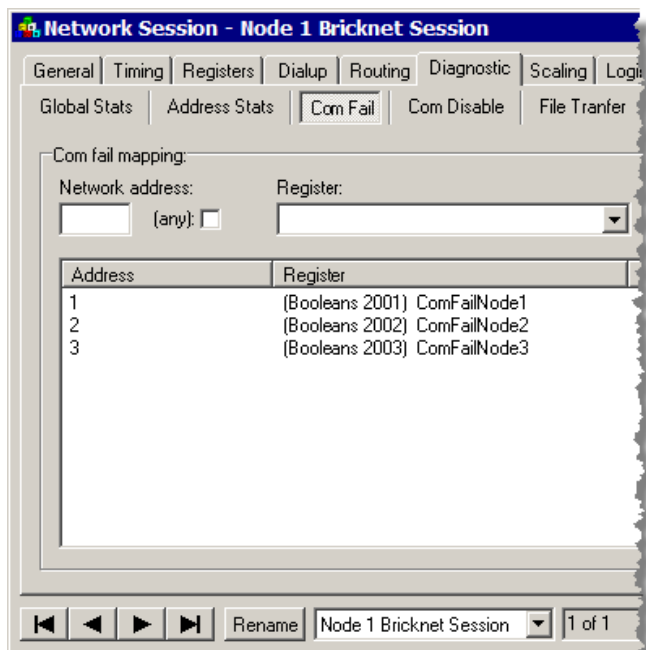
Depending on the protocol, specifies the destination node within the ScadaBuilder project. This way, the Network Event can show registers configured in the remove unit for configuring read and write events. Protocols that use this are Bricknet, SDX and STM.

For SDX and STM, you may also specify an "I/O Module" as the destination. In this case, the Network Events will show "Object" types for read and write events. The object types are known data types ie boolean, 16 bit and 32 bit integers, floating point numbers and string messages so there is no setting up data types. The protocol simply knows what it needs to transfer.

IP Address

Protocols such as Modbus TCP/UDP Master, SDX, Ethernet IP Master and DNP 3 Master require a destination IP address to identify a remote unit on an Ethernet network. If the IP address is outside of the local network, the Gateway must be configured in the **Node | Settings | Ethernet** tab.

Address Com Fail



This section sets up monitoring each address in a single network for communication failure.

Enter the Slave you want or click that (any) check box and select the boolean register to map to that comfail status.

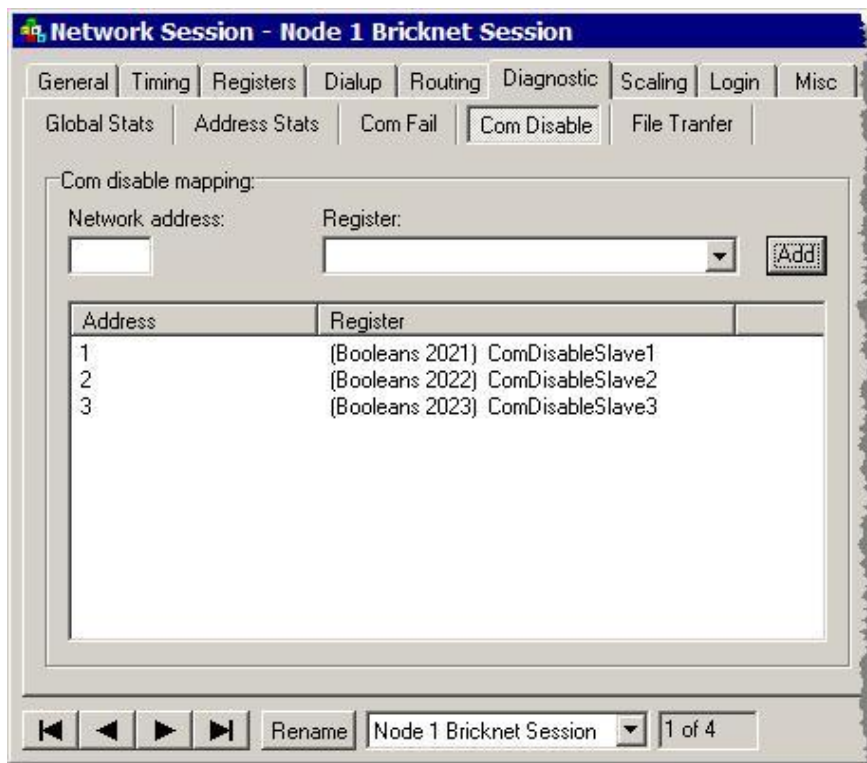
Click Add.

Com Disable

The Com Disable feature allows the disabling of remote RTU's and Controllers for either troubleshooting or system configuration purposes. A boolean register may be assigned to any address. Setting the boolean to TRUE will disable the address applied.

This feature applies to any master or initiating protocols. Such protocols currently supported are:

- Modbus RTU Master
- Modbus TCP Master
- Modbus ASCII Master
- DF1 Master
- DNP3 Master
- HART
- Bricknet



Com Statistics

All statistics that are relevant to the Network Destination are mapped as a block. These statistics provide a means of individually monitoring communication with a particular Destination. Statistics for multiple network addresses may be mapped. Address communication statistics only apply to master and peer to peer data protocols that can initiate events.

Statistic	Register
transmit command	(Integers 10003) XmtCmd_216
receive response	(Integers 10004) RcvRsp_216
receive timeout	(Integers 10005) RcvTmo_216
bad content	(Integers 10006) BadCnt_216
configuration error	(Integers 10007) CfgErr_216
lost connection	(Integers 10008) LstCnt_216
success percent	(Integers 10009) Percnt_216

Com Statistics Register

Specifies the starting register which the communications statistic is mapped to. All statistics that are relevant to the session's protocol are mapped as a block.

Com Statistics Select

This allows the selection of statistics that will be applied as a block to contiguously numbered registers. Only the statistics that apply to the protocol selected in the Network Session will be shown. Statistics may be checked or unchecked as desired. The selections are not stored between ScadaBuilder development sessions.

To exit the selection dialog, click the "X" in the upper right corner of the checkbox window.

Click the Map button to save the checkbox configuration. The results will show up in the com statistics list.

Com Statistics List

Lists the address communication statistics that have been mapped to registers. Once a block of statistics has been mapped to a particulate address, individual statistics may then be deleted from the list. All statistics for a given address must be deleted from the list before it is allowed to be re-mapped.

Routing Tab

Used for SDX serial only, each SDX serial device is automatically a router. To reach an SDX destination, one or more addresses are added to this table. The message is then transmitted with the routing information in the order given. When the remote responds, the same routing table in reverse is applied to the response message. Each unit in the path will modify the message as it goes across the link send to the next unit in line. Remember that the more routing entries there are, the longer it will take for the message to get across the routed link.

Retained Integer Registers may be used to configure the routing table on-the-fly as well.

The medium must be able to support peer to peer communications. For example:

Freewave radios have a master/slave configuration that will not support this functionality--any message routing must be handled in the radios themselves.

Digi radios are peer to peer and have no master/slave relationship so they can support routing of this type.

Steps

Enter an SDX address or Retained Integer Register for an SDX routing step.

Dialing Tab

Use for STM (SDX over cellular Text Messaging). Phone number of remote Cell Modem to communicate to a remote STM device. This can also be mapped to a Retained Message Register to allow it to be changed on-the-fly.

Number/Buffer

Enter a phone number or a Retained Message Register for the remote cell modem.

Modbus Protocol



See *The ScadaBuilder Hierarchy* (on page 73)

Due to its simplicity and wide acceptance by thousands of equipment vendors, Modbus is still one of the most commonly used SCADA protocols. Modbus is a Master/Slave protocol. A single Master communicates with up to 255 slaves. Slaves do not send messages on their own; they can only respond to messages from the Master. Modbus is designed to operate over serial networks; RS-232 for short point-to-point connections, RS-485 for longer distance hard-wired multi-drop networks, and radios and modems for even longer distances. The TCP/IP version of Modbus can operate over Ethernet and the Internet.

There are four forms of the Modbus protocol; Modbus RTU, ASCII, TCP and UDP. Modbus RTU is the most efficient and the most widely used form of the protocol over serial links. It differs from Modbus ASCII in that data is sent as binary values instead of ASCII human-readable characters. Modbus TCP has the extra overhead of TCP/IP, but also has the functionality of enabling simultaneous access by multiple masters. ICL controllers offer a choice of all four forms of the Modbus protocol.



Serial networks such as Modbus RTU and ASCII have only one master. Modbus TCP or Modbus UDP can support multiple virtual connections and therefore multiple Masters.

In its native form, Modbus can support two types of data: bits and 16-bit integers. Over the years, adaptations have been made by various vendors to support additional data types such as 32-bit integers and floating point numbers. Since there was no standard definition, there are now many different vendor specific implementations to handle these other data types. ScadaBuilder has data manipulation features built into its Modbus protocol handlers to support nearly any vendor's adaptation of Modbus.

Creating a Modbus Master Interface

Creating a Modbus Master Network Session

A Modbus Master requires more effort to configure than a Modbus Slave because the Master must initiate the messages that request data from the slaves or send data to the slaves. Slaves simply respond to data requests.

Plan how you want to set up your Modbus Master. In it's simplest form, the Master can continuously request data from Modbus slaves, and continuously write data to them. This is the easiest form of Master to create, but it is also the least efficient.

For example, remote outputs can be updated when output data changes instead of constantly writing the same output data over and over. This is accomplished by defining "Triggers". A single Trigger can look for a change in a block of registers. When a change occurs, the Trigger can cause a Modbus message to be sent that updates the remote outputs. Triggers can be added at any time as needed right from the Network Event dialog on the Activation Tab.



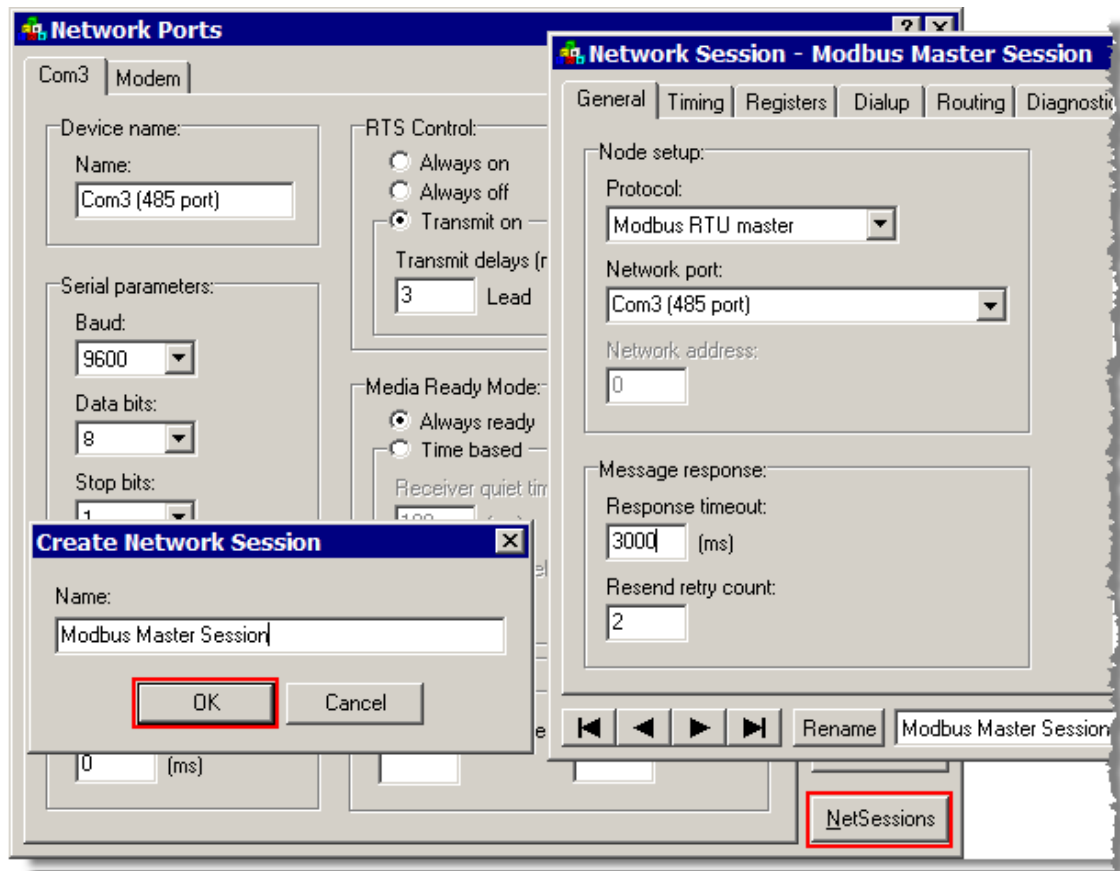
With a little more effort, it's possible to prioritize the frequency of the events that cause messages to be sent. See the Using Triggers section.

You will need to set up a Network Port that defines the basic hardware level communications elements such as baud rate, parity and hardware interface (RS-232, RS-485, etc.) or you can simply use the defaults for now and change the Network Port to your liking later on.

Some Modbus systems require dialing a telephone number via a modem to first establish a link before communicating. This is also set up in the Network Port configuration (but the phone number itself and a special modem driver may be setup in the Network Session to override the default modem driver behavior.) In addition, if you want to use the TCP/IP version of Modbus, you will need to fill in the TCP/IP information in the **Node | Settings | Target Configuration | Ethernet tab** (see "Node Settings - Ethernet Tab" on page 37) configuration.

Here we will go through the parameters that are important to a Modbus Master Session. If more information is needed on a parameter, see the *Network Session Reference* (see "Network Sessions Reference" on page 218) section of this manual.

To create a Modbus Master session, select the **Setup | Network Sessions...** menu or open a Network Port and click on the NetSessions button. A dialogue window will pop up for naming the session. Accept the default name or enter a new one. If you already have at least one Network Session, you can also simply click on the "New" button on the right-hand side of the Network Session window.



Select the protocol; Modbus RTU, ASCII or TCP/IP Master.

After selecting the protocol, choose the Network Port. ScadaBuilder only displays the port names of ports whose configuration is compatible with the selected protocol.



If you don't see the port that you want named as a choice, go back and check the Network Port configuration to make sure it is compatible. See Master Protocol List (on page 194) for compatible protocol / Network Port configurations.

Parameters of the Modbus Master Session

Response Timeout - Once a message has been sent out by the Modbus Master Session, it waits for a response within a preset period of time before giving up. If this time is set too short, then you will have poor communications. If this time is set too long, the network will get very sluggish if one or more of the slaves quits responding or if there are data errors that force messages to be resent. Generally, hard-wired links are very fast and this timeout period can be set to a fraction of a second. Radio and modem links tend to be much slower. It is not uncommon to see timeout settings of several seconds. As a rule of thumb, set this timeout to approximately twice the time that it takes for the Master message to get out and a response to be received back under normal conditions. In slower networks, remember to take into account the baud rate and maximum message length when calculating a timeout time.

Resend Retry Count - If a message is corrupted, or a slave fails to respond, the Master will try to resend the message for the number of times set in this field (typically 2 or 3). Note that a Master configured to continually read inputs and send outputs may not benefit from retries, since input requests and output commands will be repeated anyway.

Timing Parameters

Click on the Timing Tab in the Modbus Master Network Session.

Timing:

Any character timeout:
5000 (ms)

Event gap:
0 (ms)

Session gap:
0 (ms)

Probe interval:
0 (sec) ☐ Disable

Connect timeout:
35 (sec)

Session activity timeout:
30 (sec)

Probe Interval - When a slave device quits responding, it can drag down the performance of the entire network waiting for a message response timeout, yet it's important to keep trying to talk to the slave to determine when it comes up again. A Modbus Master will use the probe interval to occasionally attempt to communicate with the slave, but not as frequently as if it was up and operational. This way, the slave is checked periodically, but the impact of a "dead" slave on throughput is reduced. To be effective, the Probe Interval Time should be significantly longer than the Response Timeout Time. Note that a value of 0 makes the retries happen at the same rate as the Timeout Time. Recovery is faster this way but the system will timeout and repeat retries every polling cycle slowing the entire system down.

Probe Interval Disable - This will remove probing altogether. In order to reestablish communication with a remote, the associated Network Events must be retriggered. When the remote goes into "com fail", that unit is taken off the polling list if no Network Events have been triggered.

The next two parameters are only used if your Modbus Master is configured for Ethernet or to dial out over a public telephone link for communications. If you're not using an Ethernet or dialup connection, skip these fields.

Connect Timeout (Sec) - After dialing out to establish a link, the modem carrier must be established (dialup) or a TCP/IP connection established (Ethernet) within this time, or the Master will abort the connection.

Session Activity Timeout (mS) - If no messaging activity has occurred for this time period, the session will be disconnected (modem hangs up for dialup, or socket disconnects for Ethernet).

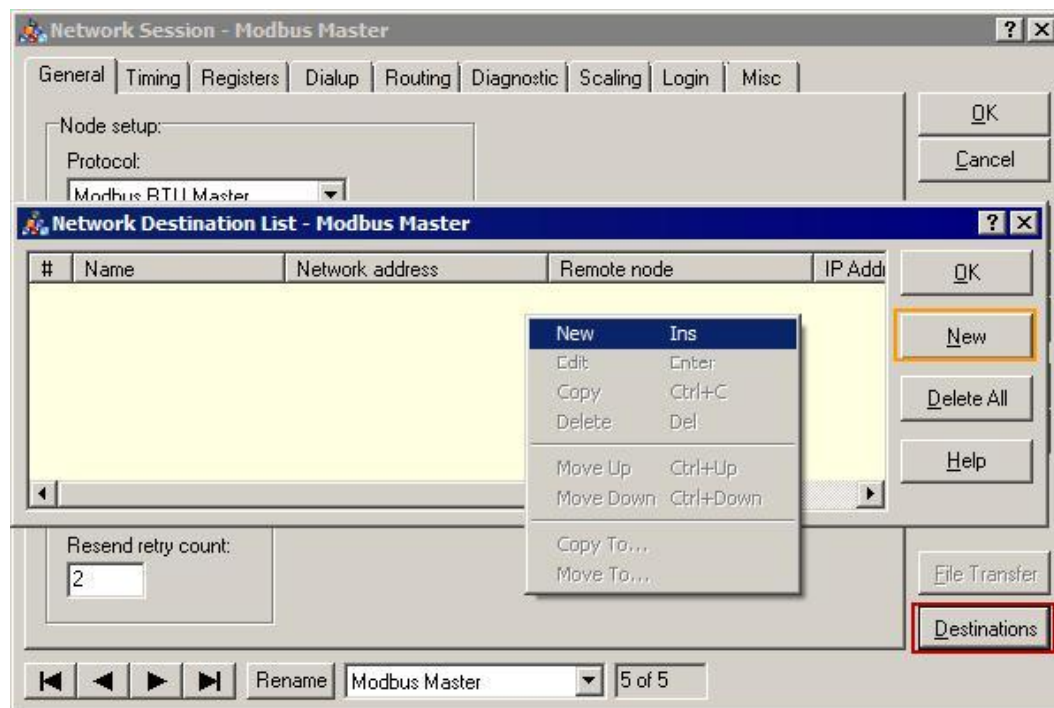
For more details, see *Network Sessions Reference* (on page 218) section.

Creating a Modbus Master Destination

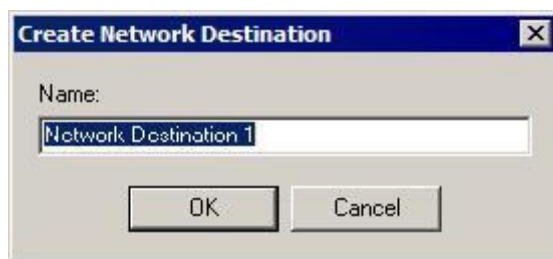


See *The ScadaBuilder Hierarchy* (on page 73).

Each remote slave that the Modbus Master Network Session needs to talk to must be setup as a Network Destination. To create a Network Destination, Open the Modbus Master Session and click on the Destinations button in the lower right hand corner to bring up the Network Destination List.



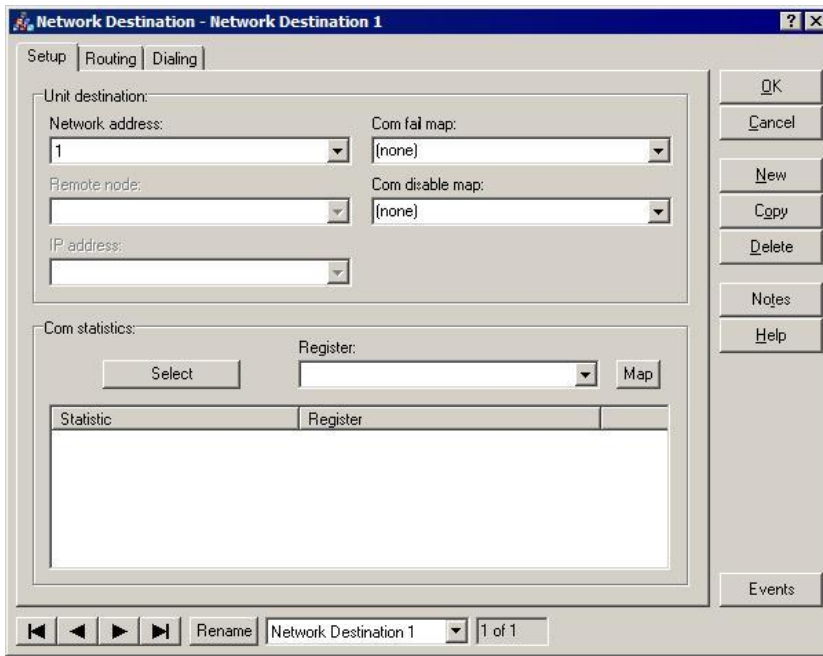
Right click in the list area or click on the New button to start a new Network Destination. You should see this dialog:



Use the default name or give it a new one. You can name it anything you like. If you do use a name it is recommended that the Site name be used and the Modbus Slave address be included so the Destination is readily identified.

Click Ok .

At minimum, enter the Network Address and if using Modbus TCP/UDP, enter the IP address of the remote Modbus slave.



Optionally, you can also map the Com Fail flag to a boolean registers to monitor the general health of the slave. You can also map the Com Disable boolean and communications statistics for more control and monitoring capability. See *Address Com Statistics* (see "Com Statistics" on page 247) for more information.

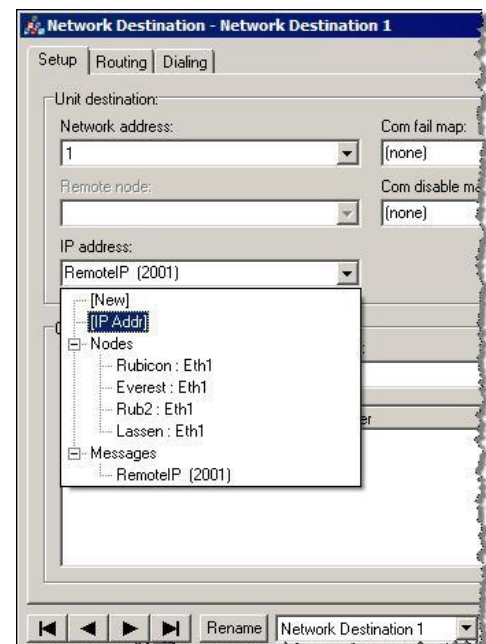
Modbus TCP/IP and UDP Master Destinations

Modbus TCP Master and Modbus UDP Master Sessions use the Network Destination a little differently.

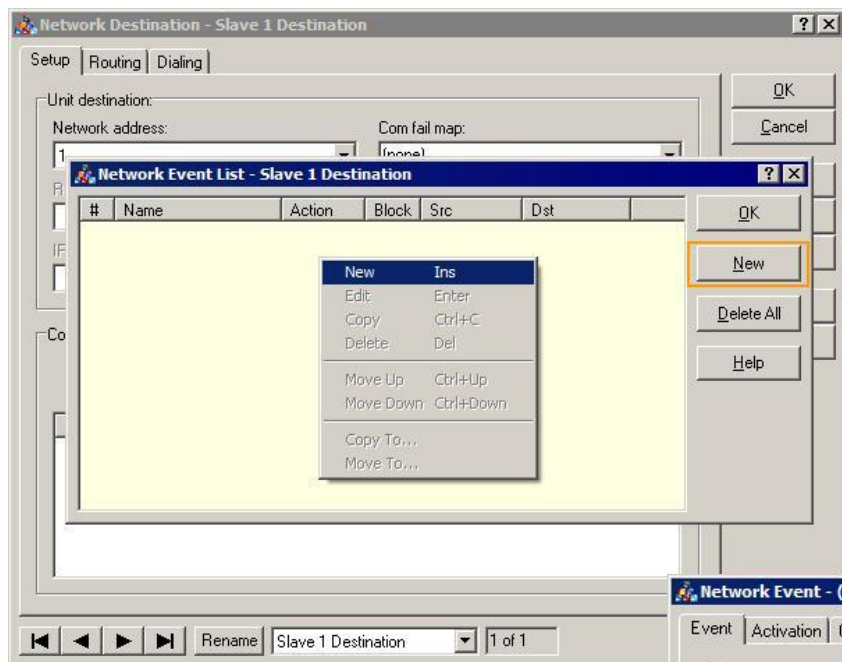
For this kind of destination one more parameter is needed: a destination node or IP address. You can select any TCP/IP enabled node in the project from the IP Address dropdown. You can also put in an IP address hard coded, or in a retained Message buffer.

The IP address uniquely identifies the remote side of the link.

Note: The Network Address field must still match the remote's Modbus slave address for the data transfer to work.



Creating Modbus Master Events



easier to understand what the events do in the future but is not functionally required by the system.

First, define the type of data transfer. An event can read registers, write registers, or probe a slave. Fill in the remote slave address to exchange data with and select an action (read, write or probe).

For efficiency, data transfers are done in blocks of multiple registers as set in the “Block size” field.

For a “read” event, data will be transferred from the slave’s Discrete Inputs or Analog Inputs to the registers in the controller. Select the Source data type and Index (register number) in the remote slave and the FIRST register of the block that the data is to be transferred to in the controller.

For a “write” event, the Source will be the first register of a block of registers in the controller and the destination will be the corresponding first register in the remote slave that the data will be transferred to.

Probe events are used to verify communications much like a ping command under TCP/IP. RTCGet and RTCSet allow the reading of or writing to the real time clock of another ICL controller. See *Real Time Clock Network Events* (on page 267) for more information.

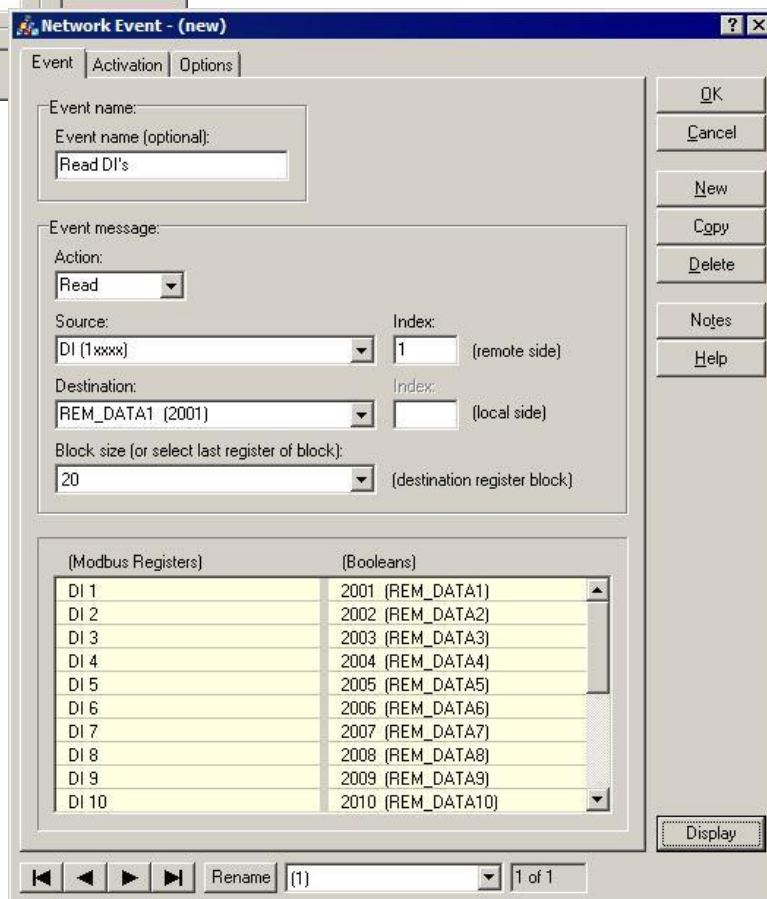


See *The ScadaBuilder Hierarchy* (on page 73).

Once a basic Network Session and Network Destination have been set up, you’re ready to set up the Network Events for your Master.

Click on the “Events” button in the lower right-hand corner of the Network Destination window. This will bring up a “Network Event List” window.

To create an event, click on “New” or right click in the list area and select New. The resulting Network Event window has an optional field for the Event name. Naming your events makes it



Click on the activation Tab configure when this event will be triggered.

See *Network Event Activation* (on page 259) section for details.

Network Events Reference

The Network Event defines a message that reads or writes information over the network. A Network Event can be Triggered (activated) in a variety of ways, either specified within the Network Event itself, or by referencing an external Trigger. This configuration tells what data to transfer to or from a remote Network Destination.

Event Name

This optional setting associates a name with the Network Event. This can be useful in organizing and tracking Network Events, especially if you have a large number of them.

Action

Specifies the action to take when the event is triggered. Read gets information from the remote unit. Write sends information to the remote unit. Probe sends a probe message to determine if the remote unit can be communicated with.

Block Size

The number of registers to read or write. (Not used for Probe type).

Source

Specifies the data source for a Read or Write operation. (Not used for Probe operation).

The interpretation of the Source depends upon the protocol being used and whether a Read or Write is being performed:

Modbus Master The ScadaBuilder Node you are setting up is a Modbus Master.

Read Set the Source to a register type. This determines the type of information read from the slave. The Index will select a specific slave register.

Write Set the Source to an allocated/named register within the current ScadaBuilder Node. This indicates where the information should come from on the master.

The ScadaBuilder Node you are setting up is a DF1 Master.

DF1 Master

Read Set the Source to a file number on the remote unit. You can select one of the predefined file numbers or enter a numeric value. The Element number selects the specific register associated with the file.

Write Set the Source to an allocated/named register within the current ScadaBuilder Node. This indicates where the information should come from on the master.

BrickNet	
The ScadaBuilder Node you are setting up is a BrickNet peer.	
<i>Read</i>	Set the Source to an allocated/named register on the remote Node. In order to get a list of remote registers, you must first select the desired Remote Reference Node Name. After you select the Source, you will be able to select a Destination register from a list of registers of compatible type.
<i>Write</i>	Set the Source to an allocated/named register on the local Node. After you select the Source, you will be able to select a Destination register from a list of registers of compatible type on the remote Node. In order to get a list of remote registers, you must first select the desired Remote Reference Node Name.
DNP3	
The Scadabuilder Node is setup as a DNP3 Master	
<i>Read</i>	Set the source registers to one of several data types. Bit data types may only be used with boolean registers while float and integer register type may be used with the Real and Integer data types in ScadaBuilder.
<i>Write</i>	Set the Source to an allocated register on the local Node. The choice will be qualified when either the Display button or the OK button are clicked. Incompatible data and message types will not allow completion of the event.

Source Index/Element

The Source Index/Element selects which register is being read from the slave unit.

The interpretation of the Source Index/Element depends upon the protocol being used and only applies to 'Read' events.

Modbus / DNP3 Master

The Source Index specifies the register index in the message.

DF1 and Ethernet IP Master

The Source Element specifies the element number to access with regards to the associated source file number.

Destination

Specifies the data destination for a Read or Write operation. (Not used for Probe operation).

The interpretation of the Destination depends upon the protocol being used and whether a Read or Write is being performed:

Modbus Master / DNP 3	
The ScadaBuilder Node you are setting up is a Modbus or DNP3 Master.	
<i>Write</i>	Set the Destination to a register type. Along with the Destination Index, this determines which register will be written on the slave unit.
<i>Read</i>	Set the Destination to an allocated/named register within the current ScadaBuilder Node. This indicates where the information should be stored on the master.

DF1 Master and Ethernet IP Master

The ScadaBuilder Node you are setting up is a DF1 or Ethernet IP Master

Write

Set the Destination to a file number on the remote unit. You can select one of the predefined file numbers or enter a numeric value. The Element number selects the specific register associated with the file.

Read

Set the Destination to an allocated/named register within the current ScadaBuilder Node. This indicates where the information should be stored on the master.

SDX and BrickNet

The ScadaBuilder Node you are setting up is a BrickNet peer.

Write

Set the Destination to an allocated/named register on the remote unit. In order to get a list of remote registers, you must first select the desired Remote Reference Node Name. After you select the Source, you will be able to select a Destination register from a list of registers of compatible type. For SDX, Remote I/O module data types can be used as well.

Read

Set the Destination to an allocated/named register on the local unit. After you select the Source, you will be able to select a Destination register from a list of registers of compatible type on the remote unit. In order to get a list of remote registers, you must first select the desired Remote Reference Node Name. For SDX, Remote I/O module data types can be used as well.

Destination Index/Element

The Destination Index/Element selects which register is being written to on the slave unit.

The interpretation of the Destination Index/Element depends upon the protocol being used and only applies to 'Write' events.

Modbus / DNP3 Master

The Destination Index specifies the register index in the message for the destination node.

DF1 and Ethernet IP Master

The Destination Element specifies the element number to access with regards to the associated destination file number.

Remote Node Name

Applies to BrickNet and Modbus TCP master only. Select the name of the Node which will be loaded on the remote unit. This will make its list of registers available for selection as Source and Destination for Read and Write Network Events.

Remote Host Address

Applies to Modbus TCP master only. This allows the IP address of the remote host to be explicitly specified. Toggle the associated "IP" check box to change the entry mode.

Event

Specifies the conditions under which the Network Event will be activated:

Startup	The Network Event will be activated once on Node startup.
Cyclic	A Cyclic Network Event is activated whenever the Cycle Count is reached. A cycle is complete when every pending Network Event for the same device has been serviced. A Cycle Count value of 5 would cause the Network Event to be activated every 5 cycles.
Timer	A Timer Network Event is activated every time the specified number of seconds passes.
Trigger	The Network Event will be activated based on a Trigger.
Manual	The Network Event will be activated "manually" under Node program control. This applies only to ScadaBuilder and C/C++ language integrated Nodes.
Com Write (self)	The Network Event will be activated whenever one of the registers specified in the source block is written to through communications in the selected Network Session. Only the register(s) that have been written to will be sent out in the event message (they will be grouped in blocks when able). This only applies to write events.
TUI Write (self)	The Network Event will be activated whenever one of the registers specified in the source block is written to through the TUI (Textual User Interface). Only the register(s) that have been written to will be sent out in the event message (they will be grouped in blocks when able). This only applies to write events.
RTC Write	Detects and triggers the Network Event when the local clock is written to from the system.

Cycle Count

For Cyclic Network Events, this parameter determines how frequently the Network Event should be activated. For instance, if Cycle Count is set to 1, the Network Event will be activated every cycle. A value of 6 will cause the Network Event to be activated every 6th cycle. A cycle is complete when every pending Network Event for the same device has been serviced.

Time

This parameter determines how frequently a Timer Network Event should be activated.

Network Session Name

For Com Write Events, this is the name of the Network Session that is used to self-trigger the Network Event.

Register Mode

This specifies how the bytes of a register value are ordered when transferred in a message. The ordering applies directly to the specified (or default) data pack type that is used to transfer values in a message.

The values are as follows:

- (default) Use register mode specified in the network session.

- Auto Converts integers (not reals) to unsigned 16-bit registers
- Auto Signed Converts integers (not reals) to signed 16-bit registers
- MSB first MSB stored in lower address, Motorola style.
- LSB first LSB stored in lower address, Intel style.
- InTouch Use Wonderware® InTouch® style.
- FIX Use Intellution® FIX® style.

Message Data Packing

Used for Modbus Network Sessions only. Sets the data type used in the message to transfer register values. A type conversion will be performed between the local register bank type and specified message data type when message transactions occur.

In Progress Map

A boolean register may be assigned showing when the Network Event is being processed. This is particularly useful when communication seems sporadic. The boolean will stay true longer when the Network Event is timing out making it easy to pinpoint trouble sites.

Command Mode

DF1 Master

Sets this Network Event to use unprotected reads and writes to a DF1 Slave. Unprotected writes can be dangerous to your remote PLC should you write to the wrong location.

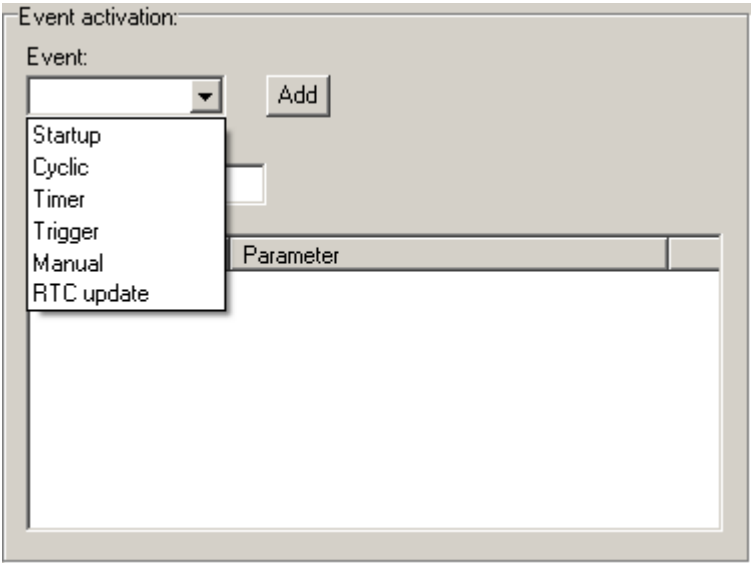
DNP 3 Master

Don't use for class and unsolicited response lookups. If this is unchecked, a DNP 3 slave may use this event as a "class" of data and write to that configured data according to the correct data type.

If the box is check, the unsolicited interface from the slave is not available.

Network Event Activation

Now it's time to define the conditions that will cause a data transfer by clicking on the "Activation" tab. An Event can have one or more activating conditions:



Startup - trigger the event once when the program first starts.

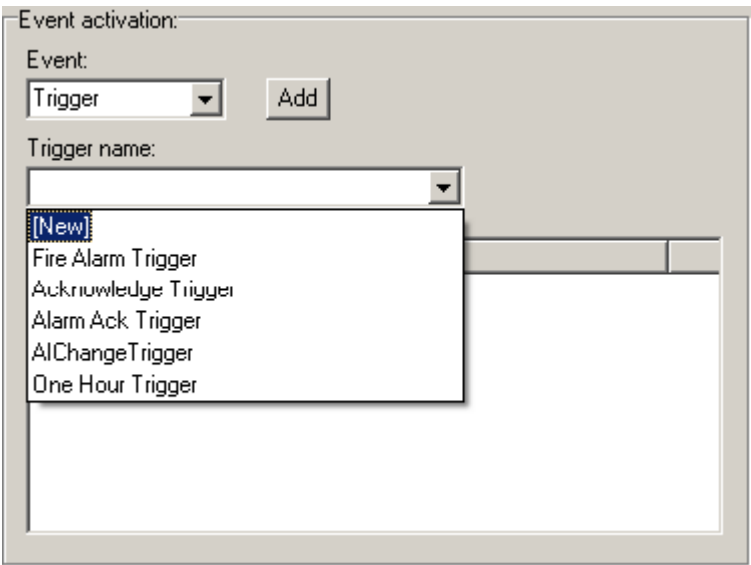
Cyclic - Continuous cycle, "round robin". Enter a cycle count (i.e. 2 means every other cycle).

Timer - Based on time in seconds.

Trigger - Select a predefined Trigger (see Triggers under Project Manager).

Manual - Manually trigger from a control program.

RTC - The Real Time Clock has been reset (used for RTCSet Network Events which are not applicable to all protocols).



If Trigger is the selected activation, a new one can be created from here by selecting the Trigger Event activation and [New] in the Trigger drop down.

To add an activation to the list, select the Event activation and, if needed, a corresponding parameter value. Click on the "Add" button. To remove an activation, right-click on the activation type in the list and "Delete".

Under the “Options” tab are a couple of selection fields to manipulate byte ordering and do type conversion as data is being transferred. This accomplishes, on a message by message basis, what can also be defined globally for the session under the Registers tab (see Network Sessions - **Registers Tab** (on page 225)).

The In progress map functionality allows a boolean register to be mapped to this event. Whenever the event is pending (currently active in the polling cycle) it will be set to true, then cleared once the Network Session moves on to the next event

The 'Options' tab contains two main sections:

- Message register overrides:**
 - Register mode: (default)
 - Msg data pack: (default)
- Register mapping:**
 - In progress map: (none)

#	Name	Addr	Action	Block	Src	Dst
1	Read DIs	1	Read	20	DI 1	REM_DATA1

Buttons: OK, New, Delete All, Help

After the parameters of the Network Event have been entered, click OK on the far right to get back to the Network Events List:

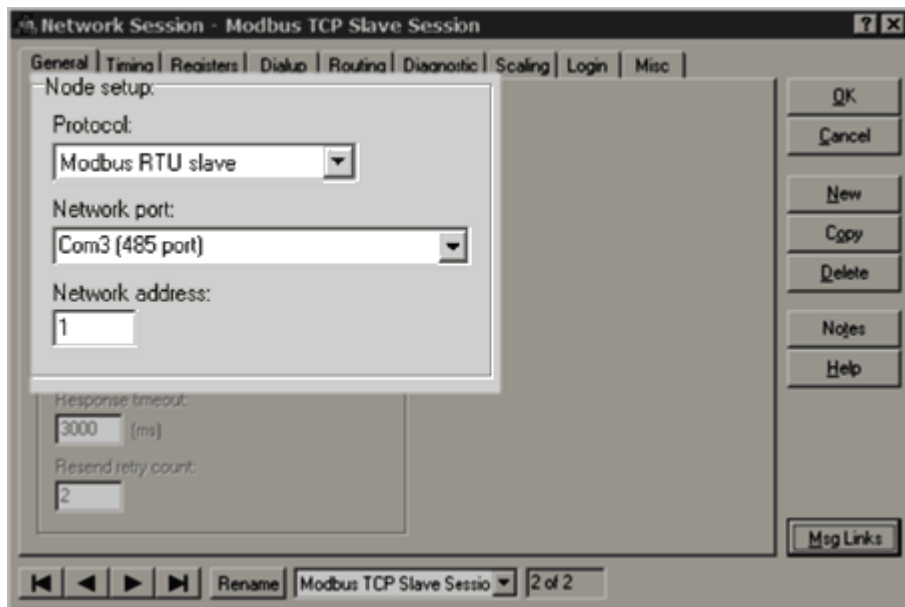
Activation Add Button

This button adds an activation condition to the Network Event. More than one condition may be added.

Creating a Modbus Slave Interface

A Modbus Slave session is easier to configure than a Modbus Master because a Slave simply needs to listen and respond to requests and commands from the Master. There's no event configuration!

To create a Modbus Slave session, select **Setup | Network Sessions...** menu. A dialogue window will pop up for naming the session. Accept the default name or enter a new one. If you already have at least one Network Session already, you can also simply click on the "New" button on the right-hand side of the Network Session window.



Protocol - Once you have a new Network Session window, select the protocol; Modbus Slave (RTU, ASCII or TCP).

Network Port - After selecting the protocol, choose the Network Port. ScadaBuilder only displays the port names of ports where the configuration is compatible with the selected protocol. If you don't see the port that you want named as a choice, go back and check the Network Port configuration.

See **Master Protocol List** (on page 194) for protocol and port support for each protocol.

Network Address - A Modbus Slave must be assigned an address between 1 and 254 (ScadaBuilder allows going all the way up to 254 even though the "official" Modbus specification does not go up that high).



You also have the option of using DIP switches on the Controller (EtherLogic or ScadaFlex Plus family) or a register for changing the slave address. This is accomplished through fields under the "Misc" Tab. If "Register" is chosen, you must select the tag name of an integer register and configure it as a "retained" (nonvolatile) register.

Download it to the controller and your port is ready to be a Modbus Slave.

Network Event and Network Message Link Display

Network Event Display

The Display control in Network Events and Network Message Links can be extremely useful to the programmer and troubleshooter. To use it in the Network Event simply click the display button after the Source, Destination and Blocksize fields are filled in.

In this example, we are reading 20 DI's from a Modbus device and placing them into REM_DATA1 through REMDATA20 (not seen). To see the bottom of the list, use the scroll bar on the right.

In Modbus, depending on whether we are reading or writing, will show one of 4 banks or data types in the Modus Registers column.

Type	Usage	ISaGRAF	Read/Write
DI	Digital Inputs, Status Registers, or 0xxxx	Booleans or Boo	Read-Only
DO	Digital Outputs, Coil Registers, or 1xxxx	Booleans or Boo	Read/Write
AI	Analog Inputs, Input Registers, or 3xxxx	Integers (ANA) or Reals	Read-Only
AO	Analog Outputs, Holding Registers, or 4xxxx	Integers (ANA) or Reals.	Read/Write



The Display Feature is particularly useful for Non Standard data types under Modbus.

Left is an example of a Session that is setup as Register Mode MSBFirst:

This Network Event is asking for 4 32bit floating point numbers and placing them in local registers.

Notice the applied offset. We are really reading from the protocol 40099 through 40106 and packing them as 32bit floating point numbers into Registers Real1 through Real4. The 40099 registers is what the message is really asking for, no other offset is applied.

Event | Activation | Options

Event name:
Event name (optional):
Read Reals

Event message:

Action:
Read

Source: AI (3xxxx) Index: 50 (remote side)

Destination: Real1 (3061) Index: (local side)

Block size (or select last register of block):
4 (destination register block)

(Modbus Registers)	(Reals)
AI 99	3061 (Real1)
AI 100	
AI 101	3062 (Real2)
AI 102	
AI 103	3063 (Real3)
AI 104	
AI 105	3064 (Real4)
AI 106	

Network Message Links Display

Network Message Link is opposite of the Network Event in that rather than getting and translating data, the Network Message link waits to get polled for data and then presents it to the protocol as configured.

In this example we have much the same as in the above Network Event Display example. The difference however is that Message Link overrides the Network Session it resides in. The Display area shows what the "new" register map looks like (at least for this one small block.).

Message link:

Start register:

Real1 (3061)

Data pack:

(default)

Block size (or select last register of block):

4

Msg type:

A0

Msg index:

1001

Link options:

Register mode:

MSB first

Integer cast type:

signed

Access mode:

write

Msg index mode:

calculated

Msg count mode:

standard

(Reals)

3061 (Real1)	A0 2001
	A0 2002
3062 (Real2)	A0 2003
	A0 2004
3063 (Real3)	A0 2005
	A0 2006
3064 (Real4)	A0 2007
	A0 2008

Msg Registers

Reg count:

8

Byte count:

16

Display

Message link:

Start register: Data pack:

Block size (or select last register of block):

Msg type: Msg index:

Link options:

Register mode: Integer cast type: Access mode:

Msg index mode: Msg count mode:

(Reals)	Msg Registers
3061 (Real1)	AO 1001
3062 (Real2)	AO 1002
3063 (Real3)	AO 1003
3064 (Real4)	AO 1004

Reg count: Byte count:

In this example though, we change the Message Index Mode and the Message Count Mode and you can emulate many of the flow meters out there.

Here we have 1 register number (index) representing a 32-bit value. Another violation of the old 16 bit Modbus Specification; the registers are now transmitted as 32-bit registers instead of 16-bit registers.

There are hundreds of permutations for numbering Modbus registers. It is likely that the ScadaBuilder Network Session - Event - Message Link interface will handle the one you need to use to get your system communicating.

Real Time Clock Network Events

A special Network Event is used to read the Real Time Clock (RTC) from another controller or write the RTC time to another controller. Two protocols can utilize this: Modbus any flavor, and Bricknet.

To do this, create a New Network Event see ***Creating Modbus Master Events*** (on page 253) or ***Creating Bricknet Network Events*** (on page 352)

In the dialog, select the RTCSet (write your controllers's time to another) or RTCGet (read another controller's time and synchronize the local controller)

This feature allows an entire system to be synchronized to one controller. Use this in combination with the GPS time sync feature, (see GPS Overview) and you have a perfectly accurate and automatically synchronized time throughout the system.



RTCSet and RTCGet access the registers 65001 through 65006 as UTC (Universal Time Code) or GMT (Greenwich Mean Time). If a time zone and or Daylight Savings Time (DST) is applied in the Node Mapping section, then those offsets are applied to the parameters below.

Register (4x)	Time Parameter
65001	mm (1=January)

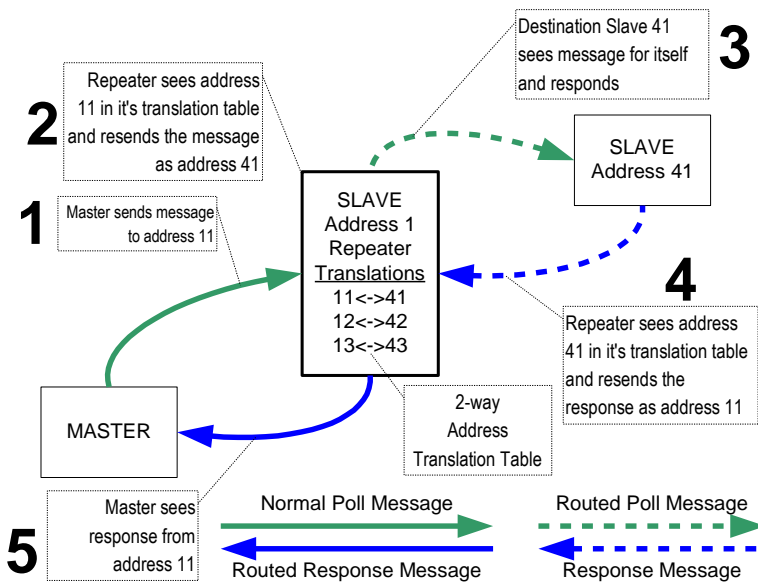
65002	dd
65003	yyyy (do not use double digit year)
65004	hh
65005	mm
65006	ss

It is recommended that the time zone and DST functions be configured before using this feature, or allowing ScadaBuilder or another controller to update the RTC. ScadaBuilder assumes UTC when it transmits the time. See **Mappings** (see "Mappings Reference" on page 493) section for more details.

Understanding Modbus Routing (Store and Forward)

Routing, also known as “Store & Forward”, is typically used in radio based systems to extend the effective range of a radio link by using intermediate stations as repeaters for messages destined for “downstream” sites.

Routing is currently supported for the Modbus and Bricknet protocols. Bricknet was designed from the ground up to support routing so configuring a set of repeaters is very straightforward. Modbus on the other hand was never designed to support repeater operation. It was originally intended for hard-wired factory floor networks, so a little addressing trickery is required.



Since Modbus was never designed with store and forward operation in mind, the Modbus messages do not have a source address imbedded in them and the Master would hear the repeated message as if it were a response, causing it to be a little confused. Because of this, an address translation scheme is required. This means that intermediate stations that are acting as repeaters have a Modbus address for each downstream station that they are repeating messages to in addition to their own Modbus address.

An example of how the Modbus repeater scheme works is pictured here.



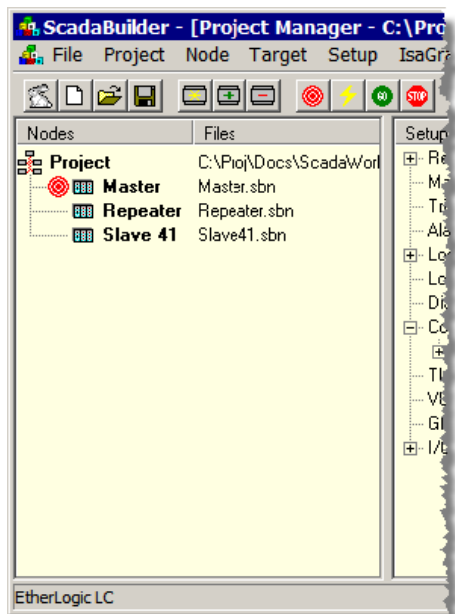
While the routing feature is the function of the Repeater Slave (Address 1 in this case), the Master in the system must be able to tolerant overhearing the Routed Poll Message.

Some Masters are not capable of throwing away unwanted messages and reject the Routed Response Message assuming there has been data corruption of some kind. Using an ICL controller as the Master will eliminate this problem.



Unlike Modbus, the SDX and Bricknet protocols have internal information in each message that supports repeater operation without going through any address translation schemes, so address translation is not used. If a system is known to need routing, it might be better to design using SDX or Bricknet on the front end.

Setting Up a Modbus Routing System



For Modbus Routing, it does not matter whether the router is in the same project or not, as a matter of convenience, we are setup with a Master, Repeater and Slave (41) in the same project.

Configure a Modbus RTU Master Session on the Master node (on whatever comport you will use for your Modbus RTU network).

Configure Modbus RTU Slave Sessions on both the Repeater (configure as slave 1) and Slave 41 (configure as slave 41).

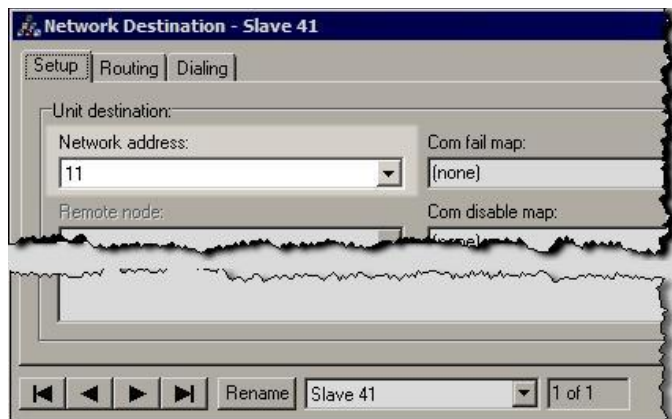
If you are not sure how to do these steps, refer to both *Creating a Modbus Master Interface* (on page 249) section and *Creating a Modbus Slave Interface* (on page 261) section.

Create the following Network Destination exactly as shown. We are configuring the system shown in the previous *Understanding Modbus Routing (Store and Forward)* (on page 268) section.

If you need help configuring Network Destinations, See *Creating a Modbus Master Destination* (on page 251) section for more details.

The Master is configured with a Modbus Master Session and events polling "Slave 41" from the Slave 41 Destination.

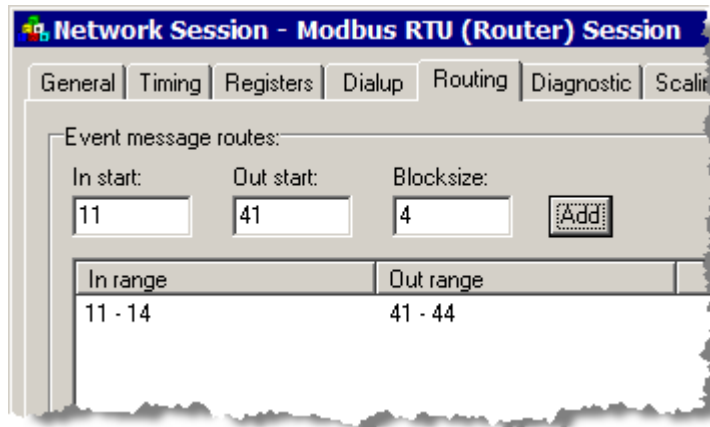
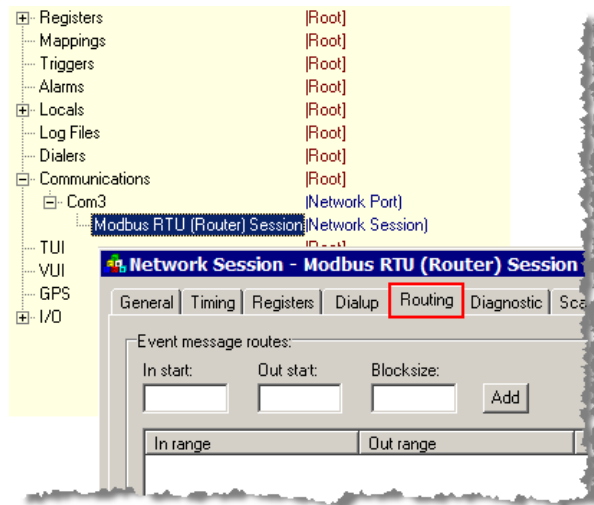
There is nothing special to do in the Master Session except to poll the routed address and not the slave address (41) directly. The Router is where all the translation work is done



Routed addresses are sometimes called "Virtual Addresses" as they do not represent a real unit but they do take up one of the 1 to 254 addresses available in a Modbus system.

Next we need to go to the Repeater's Modbus Slave RTU Session and click on the Routing tab.

The session name here is Modbus RTU (Router) Session--you can call it anything you wish but the more descriptive the name the better.



Here in the Routing dialog, we enter the routing table. This will define the block or blocks of addresses that are both "virtual" and real. We enter 11 for the "In start", 41 for the "Out start" and 4 for the "Blocksize" and click on the Add button



Address translations must go from lower addresses to higher addresses in order for the system to support all functions. This is a limitation of the Modbus protocol.

More routes may be added to this router or other routers may be added to the system (more than one repeater). The only rule is there can be NO over lapping addresses AT ALL! If a virtual address is used, you cannot use that same address for a "local" (non-repeated) slave or another router address range. It will not work. The unit that is configured for the virtual address, will try to repeat the message and cause a collision with the "real" non-repeating response.

Since there is nothing to do in the remote Slave 41 node, we are done configuring our routing system.

Download configurations to all three units and you are done.

If routing changes are needed (routed slaves added, address blocks changed) only the Router needs to be updated with the new configurations.

Using Network Message Links

If a Modbus slave is to only pass integer and boolean data per the Modbus standard, the configuration of the Slave is complete. If 32-bit integer or floating point data is to be passed, it may be necessary to perform additional configuration work to accommodate these “non-standard” data types.

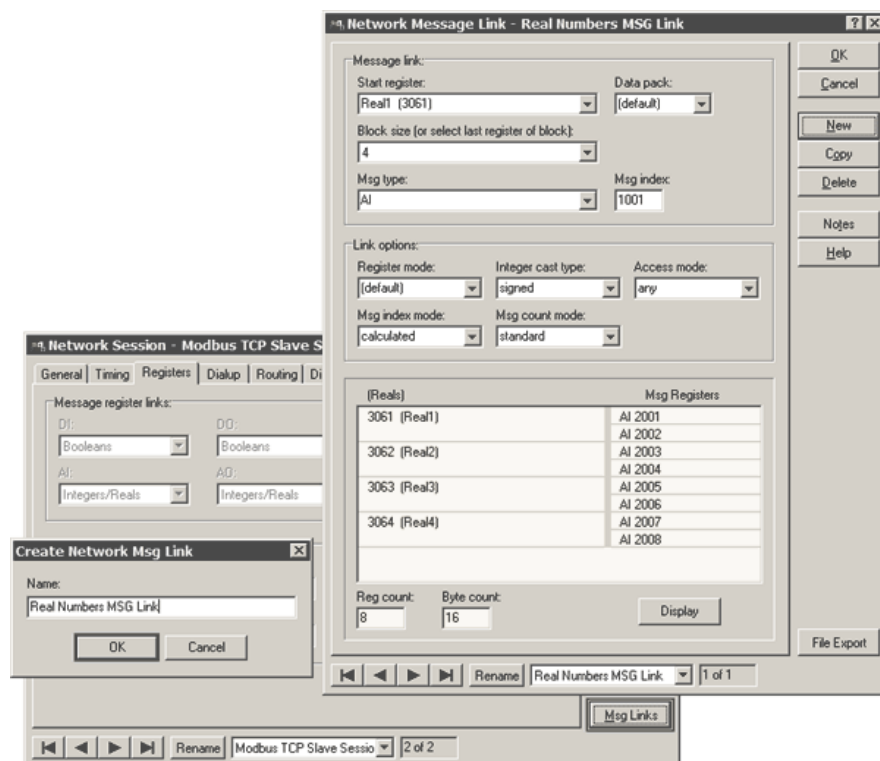
On a “global” level, 32-bit integers and floating point values can be handled with settings under the **Network Session - Registers tab** (see “Registers Tab” on page 225) as long as the byte ordering of these 32-bit data types is the same for all integer and floating point values. If the byte ordering differs based on the data type, then you must set up “Network Message Links” that will translate data formats “on-the-fly” based on the index of the blocks of data being accessed. Interfacing with InTouch - Wonderware” HMI software is an example of where this is required.

When the controller receives a message with a Modbus register number that has been linked, the associated data register will then be accessed. You can manage access to different data register types (i.e. INT16s, INT32s, floats and strings) on the same Modbus Slave session by creating individual links to the desired registers or blocks of registers.

Options are also provided to control the manner in which register data values are packed to or unpacked from a message register or string. Message start indexes may also be directly specified or automatically calculated based on the data type used to pack data into a message.

MODBUS SLAVE - Creating Network Session Message Links

If data from a Modbus Master needs translation of the byte ordering or data typing, click on the “Msg Links” button in the lower right-hand corner of the window. A dialogue window will pop up for naming the Message Link. Accept the default name or enter a new one. This will bring up a “Network Message Link” window.



In the “Message Link” area of the window, you will define the block of data that the Master will access for data transfers. In the “Link Options” section, you will define the translation functions to be performed whenever the Master accesses the defined register area.

Start Register - Select the tag name of the FIRST register in the block of registers to be accessed in the controller.

Data Pack - This sets the way data values are packed into the message when they are transferred. If “default” is selected, then the data pack type will be set to the register type of the local starting register. If the data pack type differs from the local register type, then a type conversion will be performed when the message transaction occurs.

Block Size - Specifies the number of local data registers to link.

Msg Type - Specifies the Modbus message type used for the link (AO-Modbus type 4xxxx, or AI Modbus type 3xxxx).

Msg Index - Specifies the Modbus message starting index that the local data registers are to be linked to.

Message link:

Start register: Real1 (3061) Data pack: (default)

Block size (or select last register of block): 4

Msg type: AI Msg index: 1001

Link options:

Register mode: (default) Integer cast type: signed Access mode: any

Msg index mode: calculated Msg count mode: standard

(Reals)	Msg Registers
3061 (Real1)	AI 2001
3062 (Real2)	AI 2002
3063 (Real3)	AI 2003
3064 (Real4)	AI 2004
	AI 2005
	AI 2006
	AI 2007
	AI 2008

Reg count: 8 Byte count: 16 Display

Register mode - The register mode defines how ScadaBuilder handles Modbus numeric data. There are four different modes, two that are defined specifically for use with the two most popular HMI software packages; InTouch by Wonderware and FIX by Intellution. The available modes are:

- **MSB First** - The most significant word is placed in the lowest order register followed by the least significant word.
- **LSB First** - The least significant word is placed in the lowest order register followed by the most significant word.
- **InTouch** - Compatible with Wonderware® InTouch® software.
- **FIX** - Compatible with Intellution® FIX® software.

Integer Cast Type - This field configures ScadaBuilder to interpret integers as either signed (+/-) or unsigned values.

Message Index Mode and Message Count Mode - When handling 32-bit integers and real numbers, different vendors have different ways of representing the starting register and block size in the Modbus message. Some use the 16-bit register numbers that are native to the Modbus message. In this case, when 32-bit values are transferred, each value is an odd numbered register and the even numbered register that holds the second half of the 32-bit value is skipped (1, 3, 5, . . .). Others number the 32-bit registers consecutively as single registers, ignoring the native 16-bit register numbers. Likewise, the count of the number of registers can be in terms of 16-bit or 32-bit entities.

Message Index Mode - The Message Index can be “calculated” or “direct”.

When the Message Index is set to “calculated”, the registers are always odd numbers because the starting register address is calculated from the formula **(Register Index * 2) - 1**. So for register addresses of 1, 2 and 3 in the modbus message, the calculated register numbers are 1, 3 and 5 respectively. The “calculated” mode is the more commonly used technique and is the default ScadaBuilder setting.

When the Message Index is set to “direct”, the register address is used directly from the Modbus message without interpretation.

Message Count Mode - The Message Count can be “standard” or “encoded”.

When set to “standard”, the register count in the Modbus messages refers to the number of 16-bit registers required to transfer the register data bytes. For example, if 5 floating-point registers (32-bits) are being transferred then the register count in the Modbus message will be 10. It takes 10 16-bit Modbus registers to transfer 5 32-bit floating-point registers.

The “standard” mode is the more commonly used technique and is the default ScadaBuilder setting.

When set to “encoded”, the register count in the Modbus messages is the number of registers of whatever data size that is being transferred (int16, float, etc.). For example, if 5 floating-point registers (32-bits) are being transferred then the register count in the Modbus message will be 5.

Network Event and Network Message Link Display (on page 262) section for data on the display registers window.

Network Message Link Reference

A Network Message Link allows you create a link between data registers and message registers. Message links are only used with slave protocols. These links essentially allow you to map different local data registers (or blocks) to specific Modbus register numbers.

When the controller receives a message with a register number that has been linked, the associated data register will then be accessed. You can manage access to different data register types (i.e. INT16s, INT32s, floats) on the same slave session by creating individual links to the desired registers (or blocks). Each message register number can only be linked to one data register.

Options are also provided to control the manner in which register data values are packed to or unpacked from a message. Message start indexes may also be directly specified or automatically calculated based on the data type used to pack data into a message.

Net Msg Links are also used under DF1, Ethernet IP and DNP 3 Slave protocols to tell the system where to place and get data according to data types received on the Network port. This way, register is accessible even though its network address offset may be well beyond the protocols ability to address.

See the *Slave Protocol List* (on page 196) for those protocols requiring and supporting Network Message Links.

Data Pack

This sets the way data values are packed into the message when they are transferred. If "default" is selected then the data pack type will be set to the register type of the local starting register. If the data pack type differs from the local register type, then a type conversion will be performed when the message transaction occurs.

Start Register

This is the starting point for the block of data registers that is to be linked on the local side.

Block Size

This specifies the number of local data registers to link.

Message Type

This specifies the Modbus Slave message type (AO 40xxxx or AI 30xxxx) to be used for the link.

File Number

Under DF1 and Ethernet IP Slave, this parameter becomes one of several "File Numbers".

O0	Output Image
I1	Input Image
S2	Status
B3	Bits
T4	timers
C5	Counters
N7	Integers
F8	Floating point

Of the above types, the most commonly used are the Status, Bits, Integers, and Floating point data types.

You may also specify a non-standard file number by clicking on the "Number" option and enter a numeric value (with no letter).

Group / Variation

This parameter allows the user to select the data Group that DNP3 Master will use to access the register defined in the Network Message Link.

Variations with no flags simply operate on "static" or live register data.

Variation with flags can be setup with different configurations including Freeze, Arm, and Reset features.

Group	Data Type(s)
Binary Inputs (Packed)	Booleans
Binary Inputs (Status)	Booleans
Binary Outputs (Packed)	Integers or Reals
Binary Outputs (Status)	Integers or Reals
Binary Counters (32bit w/flags)	
Binary Counters (16bit w/flags)	
Binary Counters (32bit w/ no flags)	
Binary Counters (16bit w/ no flags)	
Analog Inputs (32 bit w/flags)	
Analog Inputs (16 bit w/flags)	
Analog Inputs (32 bit w/no flags)	
Analog Inputs (float w/flags)	
Counters	Integers

Event Class

Any Network Message Link created can be assigned to a DNP 3 class (1 through 3). Just select the Data Class to apply this Message Link to. When a DNP 3 master asks for a class of data, the controller will respond with the data in every Message Link assigned to that class.

Message Index or Element

This specifies the slave message starting index that the local data registers are to be linked to.

As a DF1 element, it tells the system what position in the "file" to apply the register configured.

Register Mode

This specifies how the bytes of a register value are ordered when transferred in a message. The ordering applies directly to the specified (or default) data pack type that is used to transfer values in a message.

The values are as follows:

- (default) Use register mode specified in the network session.
- MSB first MSB stored in lower address, Motorola style.
- LSB first LSB stored in lower address, Intel style.
- InTouch Use Wonderware® InTouch® style.

- **FIX** Use Intellution® FIX® style.

Integer Cast Type

Determines if integers in Modbus messages should be treated as signed or unsigned values. This only applies when type conversions are taking place between Modbus messages and target registers of a different type.

Access Mode

This sets the access mode on the local register side for which the link applies.

- **Any** Link will be applied to both read and write messages.
- **Read** Link will only be applied to read messages.
- **Write** Link will only be applied to write messages.

Message Index Mode

This sets the mode used to determine the actual message start index for Network Events and Network Message Links that use this session.

Calculated This mode bases the message start index on the transfer data type and message index specified for the link. For example, if a message index of 5 is specified for the link and the data pack type is "float" (32 bits), the actual index transferred in the message will be 9. The resulting index of 9 is based on that 2 Modbus registers (16 bits each) are required to transfer every floating point register value (32 bits). Thus, the 5th floating point register would start at 9th Modbus register.

Direct This mode uses the message index specified for the link as the actual starting index.

Message Count Mode

This count mode is provided for compatibility with equipment that runs a variation of the Modbus protocol.

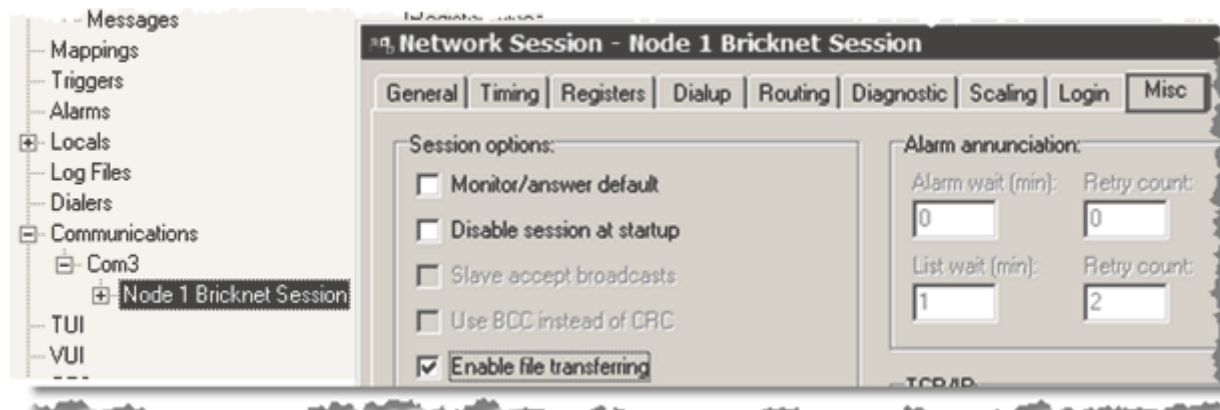
Standard This mode is the most common. The register count in the Modbus messages refers to the number of 16-bit registers required to transfer the register data bytes. For example, if 5 floating-point registers (32-bits) are being transferred then the register count in the Modbus message will be 10. It takes 10 16-bit Modbus registers to transfer 5 32-bit floating-point registers.

Encoded This less common mode interprets the register count in the Modbus messages as the number of registers of whatever data size that is being transferred (int16, int32, float, etc.). For example, if 5 floating-point registers (32-bits) are being transferred then the register count in the Modbus message will be 5. Many flowmeters use this mode with registers in the 7000-8000 range (AO or AI).

File Transferring Over Modbus And Bricknet

To transfer files serially over a Modbus or Bricknet network requires that the Sessions for both sides (the Node or Master initiating the File Transfer Event and the Node or Slave responding) have File Transferring enabled in their respective Network Session.

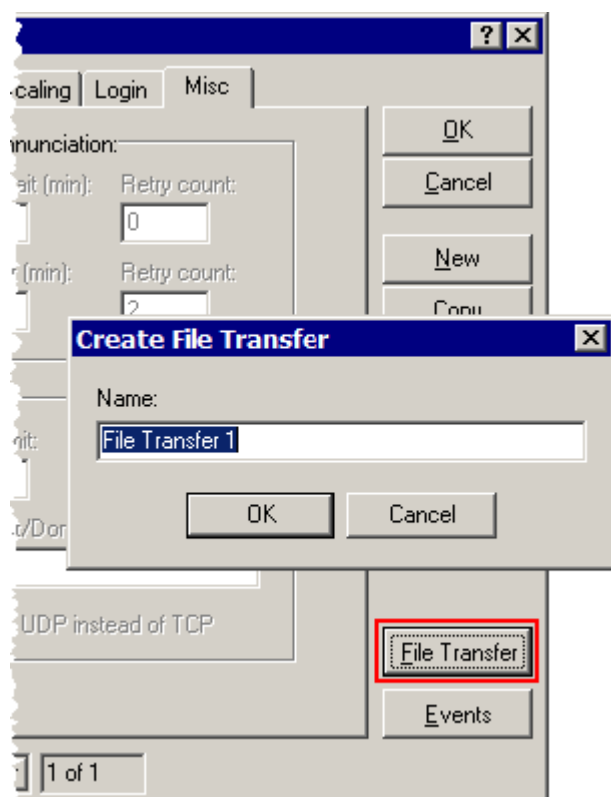
To do this, open the particular session with which you want to transfer files, click on the Misc (Miscellaneous) tab and check the Enable File Transferring box:



This must be done for every session that uses the File Transfer feature (including Modbus Slaves).



File Transferring takes some memory resources. Most of the time those memory resources will not be in use.



You're ready to configure the names of files to be transferred and the initiating Triggers and other parameters related to the data transfer.

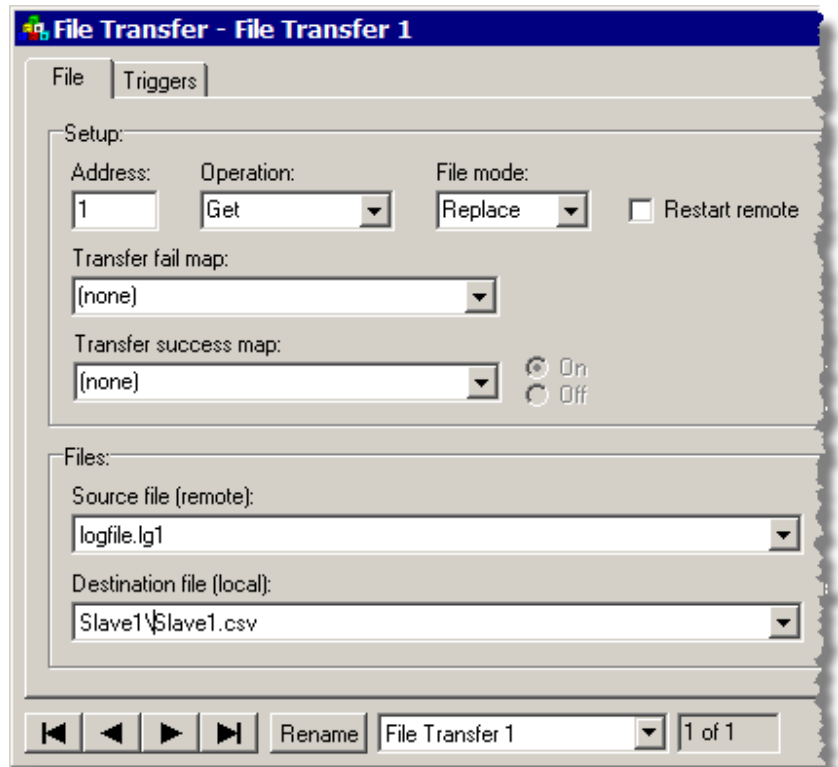
To create a File Transfer Event (Master or Bricknet), click on the File Transfer button in the lower right hand corner (after enabling file transferring.)

Operation - Specifies the transfer operation to be performed. Selecting 'Get' will retrieve a file from the server. Selecting 'Send' will send a file to the server. Selecting 'Send/Delete' will send a file to the server and then delete it from the local controller once delivery is confirmed.

Mode - Allow the programmer to configure whether the transfer will replace an already existing file or append to it.

If we are getting a file, the Destination field is the local disk on the controller. If we are sending a file, then the destination is on the remote node.

Restart Remote - check this if this File Transfer Event is downloading a new program to a controller. After the new program is downloaded successfully, the remote system will then reboot.



Triggers Tab - Any trigger can fire a File Transfer. See Using Triggers for more information.

For further information, see the following sections in this manual:

Network Session - File Transfer (see "File Transfer" on page 234), **Remote Host File Transfers** (on page 281) and **File Transfer Event Reference** (on page 279)

File Transfer Event Reference

File Transfers allow files to be read from or written to other nodes via the Modbus or Bricknet protocol. To enable the feature in a Modbus or Bricknet Session, click on the Misc tab within the Network Session and check the Enable File Transferring check box.

The feature allows for updating application programs on nodes as well as getting system or user log files. File transferring may be used over any serial media that supports the Modbus or Bricknet protocols.

This feature is part of the **Remote Host File Transfer** (see "Remote Host File Transfers" on page 281) feature in ScadaBuilder.

Event - Address

The address configures the destination node on the network that the File Transfer Event needs to get or send files. This address works the same as the Network Event Address parameter.

Operation

This specifies the transfer operation that will be performed with the remote Node.

- Selecting 'Get' will retrieve a file from the Node.
- Selecting 'Send' will send a file to the Node.
- Selecting 'Send/Delete' will send a file to the remote and then delete it from the local Node.

- Selecting 'Get/Delete' will get the file from the remote and then delete it from the local Node. The delete functions will not occur until a complete transfer of the file is accomplished.

File Mode

This specifies the method used when sending or getting a file.

Selecting 'Replace' will overwrite the destination file if it already exists.

Selecting 'Append' will add to the end of the destination file if it already exists (otherwise it will be created).

Restart Remote

This option is typically used when update one controller from a Remote Hosting master. After the file(s) have been successfully downloaded to a remote, a normal shutdown and restart will occur.



Caution should be used when restarting any Node that is doing active control. Be sure that an arbitrary restart is safe for both equipment and personnel when using this feature.

Transfer Fail Map

This specifies a Boolean register that is used to map the transfer failure status. If a File Transfer was unable to complete (after retries) then a value of '1' will be loaded into the register indicating a communications failure. When a file transfer is successful then the register will be cleared back to '0'.

This flag may be reset by the control program or Local Event at any time.

Transfer Success Map

This specifies a Boolean register that is used to map the transfer success status. The associated "On/Off" control selects the value that will be written to the register when a successful file transfer has completed (On = 1, Off = 0). It is up to the control program to reset the value in the register to the opposing state.

The reason for being able to set the flag to the off state is to reset the Archive flag after a log file has been archived. This closes the loop on an archive after it has been transferred off the Node.

Source File

This specifies the source file that will be used for the file transfer.

- When getting a file the source file is on the remote Node side.
- When sending a file the source file is on the local side.

If the file does not exist then a File Transfer error will be generated.

A hard-coded string or Message buffer may be used to specify the file name. A Message buffer allows the file name to be changed at runtime.

Directories may be specified with a "\" such as c:\logs, a trailing backslash is not needed. Forward slashes could cause errors in directory lookups. It is always better to use the "\" character to specify directory names.

Destination File

This specifies the destination file name that will be used for the file transfer.

- When getting a file the destination file name refers to the local side.
- When sending a file the source file name refers to the remote Node side.

A hard-coded string or Message buffer may be used to specify the file name. A Message buffer allows the file name to be changed at runtime.

Trigger

This allows you to select the Triggers that cause the File Transfer Event to be activated. Individually select each desired Trigger, then click the Add button.

Remote Host File Transfers

Remote Program Updates

ScadaBuilder provides the ability to transfer files from a host controller to one or more remote controllers using either the Modbus or Bricknet protocol. Files of any type can be sent or retrieved, including log files, ScadaBuilder configuration files and ISaGRAF application files. To use the file transfer feature, there are a few configuration parameters that need to be setup for both the host and remote controllers.

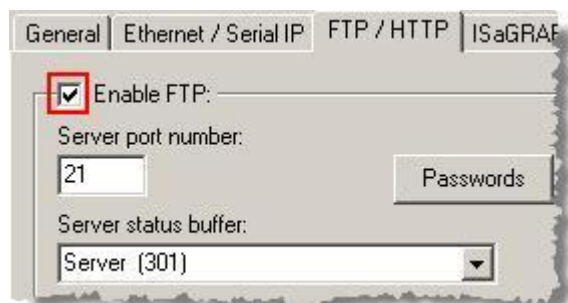
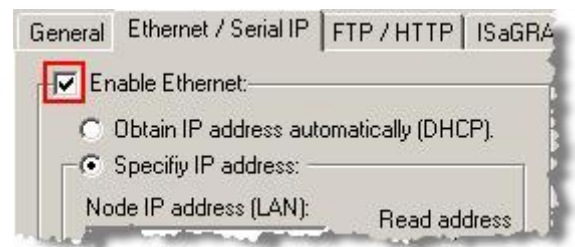


A system must start out with one Master controller and multiple remotes or in this case slaves. For purposes of illustration, there are three slaves in our example "Download" system.

The Master Configuration

Enable TCP/IP and FTP

In order for the host to transfer files to and from the remote nodes. FTP must be enabled on the Master node. In order to enable FTP, TCP/IP must be enabled first. Enable TCP/IP by clicking the "Enable Ethernet" checkbox on the Ethernet / Serial IP tab.



Once TCP/IP has been enabled, FTP must be enabled in the host node **Node | Settings | FTP/HTTP** tab dialog by clicking the "Enable FTP" checkbox.

To utilize this feature, you must have a Network Session, a Modbus Master RTU session on the Master and a Modbus RTU Slave session on each of the Slaves. See *Creating a Modbus Master Interface* (on page 249), *Creating Modbus Master Events* (on page 253) and *Creating a Modbus Slave Interface* (on page 261) for more information. These interfaces are assumed here.

If you are configuring a Bricknet system, See *Creating a Bricknet Session* (see "Creating a Bricknet Interface" on page 348). For the File Transfer interface, it is the same between Modbus or Bricknet. The only difference is that any Bricknet node can initiate a File Transfer.

Enable File Transferring

For the host node, a Modbus master or Bricknet session must be setup first. Once the Modbus master or Bricknet session is setup, file transfers can be enabled by checking the "Enable file transferring" check box in the "Session options" area on the "Misc" tab.

The Slave Configuration

In the project window, double click on Slave1 in the project window to open the Node Setting dialog.



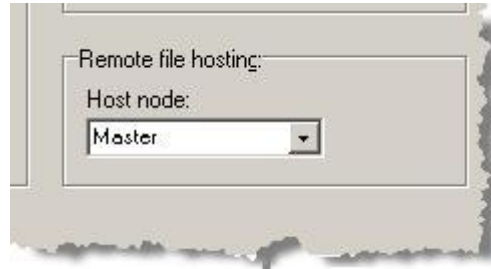
Setup the Node Address

Set default Node address in the "General" tab in the "Node Settings" dialog.



Setting Node Address parameter allows us to automatically setup File Transfer Events later on.

In the same dialog, select the file host from the drop down "Remote file hosting" list box. In this case we will use the Master node as our host.



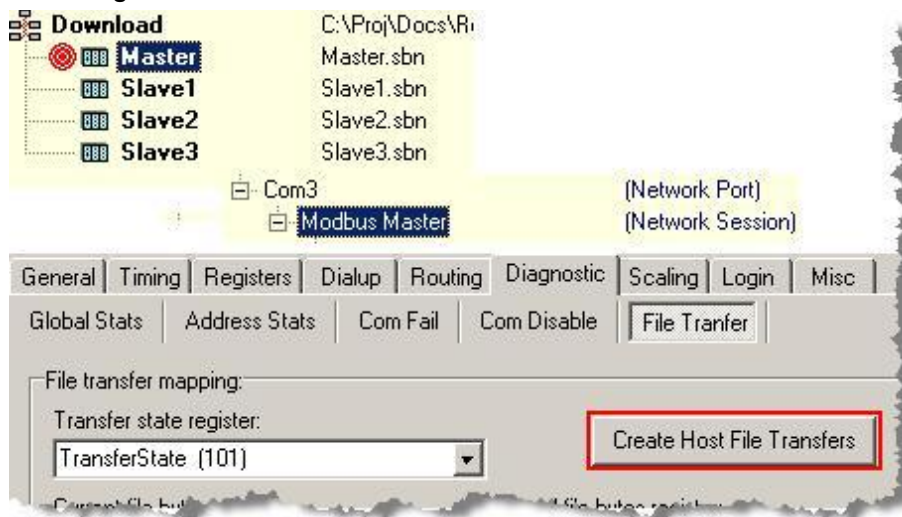
For the remote node, a Modbus slave or Bricknet session must be setup first. Once the Modbus Slave or Bricknet session is setup, file transfers can be enabled by checking the "Enable file transferring" check box in the "Session options" area on the "Misc" tab.



Repeat the Remote Node Configuration steps for the remaining Slave nodes.

The Master Host Transfer Configuration

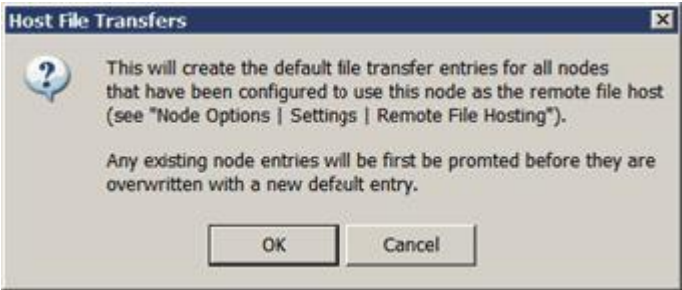
Once the remote nodes are configured for file transfers, you can easily create events for file transfers using the "Create Host File Transfers" button. This will create a directory on the host for each remote node. It will also setup a default File Transfer that will send new program configurations to the remote nodes when you use the ScadaBuilder "Send node setup to host" utility.



Open the Master Node and open the Modbus Master Session. Select the Diagnostic tab and the File Transfer sub tab.

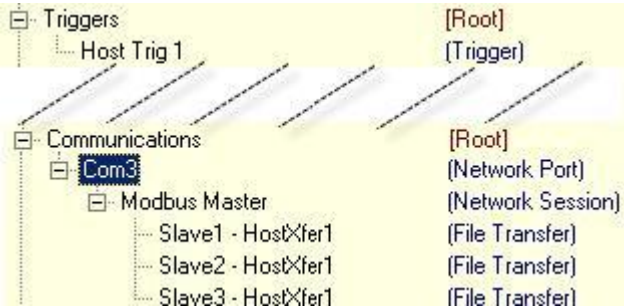
To automatically create the file transfers, click the "Create Host File Transfers" button on the "Diagnostic" tab of the master session dialog.

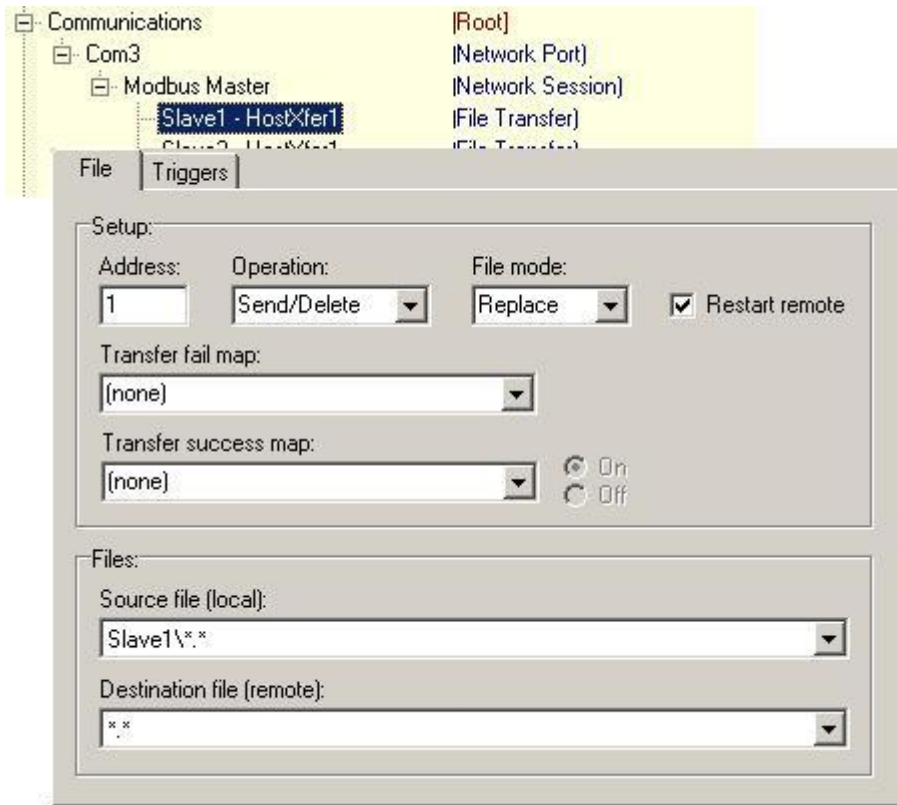
You will See this dialog.
Click OK.



The system will then prompt you for a time period. This time is how often the host controller will check for files to transfer to the remotes. Triggers are setup for you based on this value.

The configurations to the left shown condensed are the records that have been created by the Create Host File Transfers button.





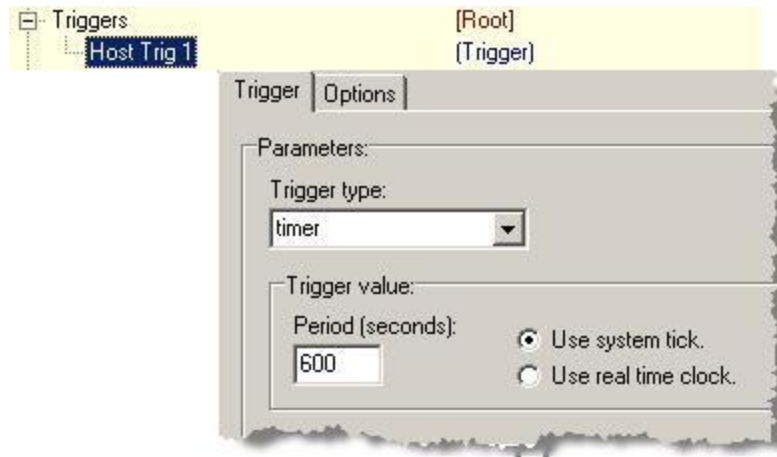
You will now have file transfers for each remote node that was configured for remote downloads. A typical file transfer configuration is shown here. This file transfer will send files to the remote node with address 1. The files will be sent then deleted from the host node and they will replace any files on the remote node with the same name after a successful transfer.

The "Restart remote" check box will cause the remote node to be restarted once the files have been successfully received. This option must be selected if you would like to download a new program and have it take effect immediately. If you do not select "Restart remote" when downloading new program files, the new program will only take effect when the remote controller is next rebooted. Make sure that you fully test the new program before you download it to the controller. If the program has any problems that prevent communication, you may not be able to download any further files to the controller.

While this transfer is setup to send all files in the Remote directory on the host, you can change it to send a subset of the files or to send files from or to a different directory. You can use the standard DOS wild card characters ? and * when specifying file names. ? will substitute for any single character and * will substitute for multiple characters.



File names must adhere to the standard DOS 8.3 format. That is no more than 8 characters to the left of the "." and no more than three characters to the right.



You will also need to setup a Trigger to cause the File Transfer to take place. The default Trigger setup by clicking the Create Host File Transfers button is shown here. Other Trigger types can be used though. A TUI button Trigger, for example, is useful for manually Triggering a File Transfer.

Download configurations to all controllers.

To utilize the function, all controllers involved with the file transfers must have their configurations downloaded to them. If this step is left out, the system will not work.

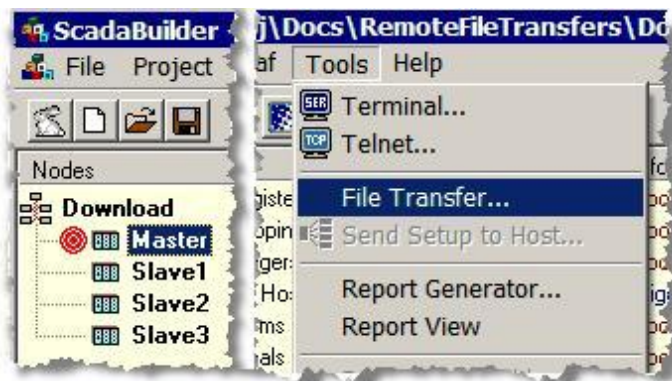
Configuring Folders on the Master

The host transfer events on the master use folders to differentiate one slave's files from another's. These folders must be created on the Master or host node. They are not created automatically. Look at the File Transfer events for each slave, there will be a Source file (local) field. The folder name before the "\" (shown below) must be created on the host node.

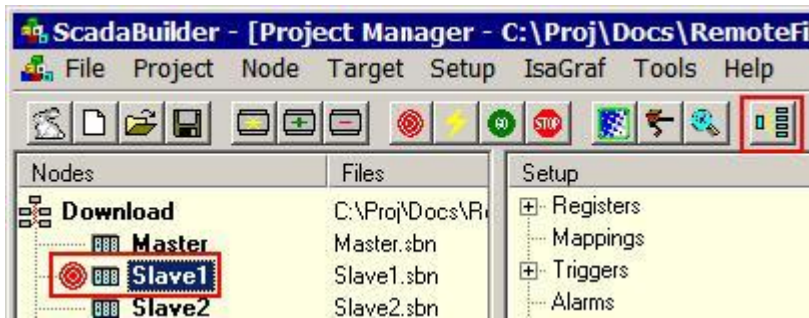


Connect to the Master Node using the **Tools | File Transfer...** menu. See *TOOLS Menu* (on page 64) for more details.

Create the slave folders for each slave



Sending remote node files to the host



Remote node files must first be transferred to the host node before they can be transferred to the remote nodes.

ScadaBuilder provides an easy method to do this. After you have made all of the changes to the remote node

ScadaBuilder configuration and ISA GRAF programs, simply click the "Send node setup to host" button or select **Tools | Send Setup to Host...**

This will transfer all of the necessary files to the remote node's directory on the host node controller. Once this has been done for each of the remote nodes, the trigger for the file transfer will initiate send the files to the remote nodes.

Secure Data Exchange (SDX and STM) Protocols

SDX Protocol

The Pinnacle series of controllers now supports ICL's Secure Data Exchange protocol otherwise known as SDX. This interface can talk over RS-232, RS-485, radio, dialup as well as Ethernet (UDP) ports. It is a 128 bit AES encrypted protocol with a 16 character user defined key. All units communicating on the same network must be configured with the same 16 character key. Up to 65535 addresses may be used in a single network.

RTU Compatibility

SDX is also fully compatible with the Ascent, Sprite, and Solaras I/O modules available from ICL. With a small amount of configuration, Pinnacle controllers may also receive quiescent messages from these remote units even though they are not programmable. The remote units may send data periodically or based on change of state or change of level.

SDX Routing

SDX messages may be routed through any SDX compatible units up to four hops (repeaters). No extra configuration is necessary as the initiating unit defines the routing information at transmission time. The "route" information is encrypted in the message and used by any SDX remote the message is addressed to. By definition, any SDX remote or controller is a router. The remotes, when receiving a routed message, will automatically turn the message around and use the same remotes to route the response back to the initiating "master".

SDX Objects, Message Efficiency, and Data Security

SDX is also capable of combining multiple data types into one message. In most traditional protocols, only one data type can be transacted at any given time. SDX is object oriented where not only is each data type an object but also each I/O type can also be an object. If there are several object types to be sent to a particular unit, SDX can combine the object types into one message eliminating the need for multiple transactions—this saves on the amount of air time consumed over a radio system for example. All this is done over AES 128 bit encryption so all of the data is safe and secure. After the data is decrypted by the AES engine with the 128 bit (16 character) key, it is checked again by two Cyclic Redundancy Checks (CRC's), one for the transport layer then one for the object layer (data content layer). Only when both of these CRC tests have passed, will the remote or master unit receiving the message begin to parse the actual object data.

STM (SDX over Text Messaging) Protocol

STM is an extension of the SDX serial protocol. With an external cell modem (Messenger Series), binary data can be exchanged using a text message transaction over a cellular network. All data types supported by SDX are supported by STM. There is no routing support for STM, it is not needed if there is sufficient cellular coverage at a site.

Creating An SDX Session

Because SDX is a peer-to-peer protocol, there are more choices on how a system is designed. It's possible to set up a Master-Slave type relationship just like Modbus; for example, a pump controller that continually polls for level information from tank sites. On the other hand, a more efficient system can be created having the tank sites send level information to the pump controller only when the tank levels change, and periodically to make sure that the Pump Controller knows that the tank sites are still alive. Because time is not wasted by constant polling, response times are faster and less network bandwidth is used; ideal for slower networks like radio based systems.

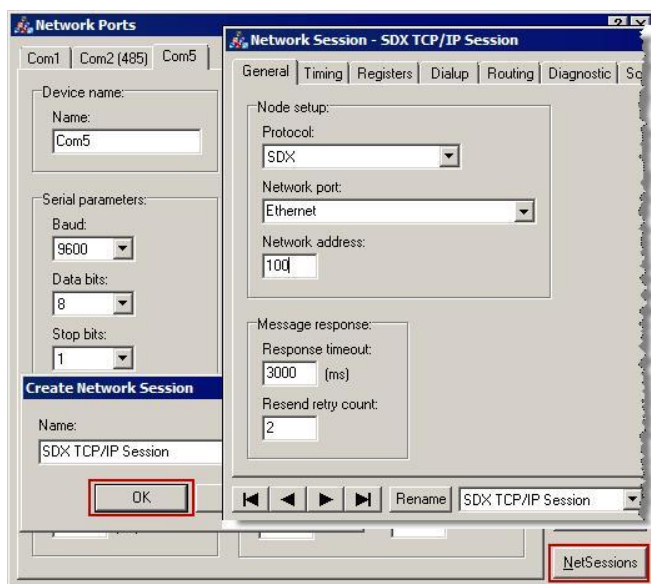
In the above example, the tank sites would use "Triggers". One of the available Trigger types senses a level change of a preset amount (Delta Trigger). When the tank level changes, the Trigger is set, causing a message to be sent to the pump station. Another Trigger might be used for an intrusion alarm, immediately sending a message when a discrete input signal from a motion sensor detects an intruder (State On Trigger). *See Using Triggers for more details.*

You will need to set up a Network Port that defines the basic hardware level communications elements such as baud rate, parity and hardware interface (RS-232, RS-485, Ethernet etc.) Some SDX (and all STM) systems might require dialing a telephone number via a modem to first establish a link before communicating. This is also setup in the Network Port configuration. You can also select the defaults and change the Network Port configuration later.

All SDX units including Controllers and RTU's (IO Modules) can initiate and receive messages in some way. You can configure your SDX Session to receive data on exception leaving the responsibility of message initiation in the remote where the data can be monitoring for change. In RTU's (IO Modules) this is usually call "Master Mode" and can function over a variety of serial and TCP/IP ports.

Here we will go through the parameters that are important to an SDX Session. If more information is needed on a parameter, see the **Network Session Reference** (see "Network Sessions Reference" on page 218) section of this manual.

To create SDX (or STM) session, select the **Setup | Network Sessions...** menu or open a Network Port and click on the NetSessions button. A dialogue window will pop up for naming the session. Accept the default name or enter a new one. If you already have at least one Network Session, you can also simply click on the "New" button on the right-hand side of the Network Session window.



Select the protocol; SDX (or alternatively STM).

After selecting the protocol, choose the Network Port. ScadaBuilder only displays the port names of ports whose configuration is compatible with the selected protocol.



If you don't see the port that you want named as a choice, go back and check the Network Port configuration to make sure it is compatible. See Master Protocol List (on page 194) for compatible protocol / Network Port configurations.

Parameters of the SDX Session.

Network Address - An SDX node address must be assigned an address between 1 and 65535. You also have the option of using a register for changing the unit address. This is accomplished through fields under the "Misc" Tab. If "Register" is chosen, you must select the tag name of an integer register and configure it as "retained" (nonvolatile).

Response Timeout - Once a message has been sent out by the SDX Session, it waits for a response within a preset period of time before giving up. If this time is set too short, then you will have poor communications. If this time is set too long, the network will get very sluggish if one or more of the units fails to respond or if there are data errors that force messages to be resent. Generally, hard-wired links are very fast and this timeout period can be set to a fraction of a second. Radio and modem links tend to be much slower. It is not uncommon to see timeout settings of several seconds. As a rule of thumb, set this timeout to approximately twice the time that it takes for the Master message to get out and a response to be received back under normal conditions. In slower networks, remember to take into account the baud rate and maximum message length when calculating a timeout time.

Resend Retry Count - If a message is corrupted, or a remote fails to respond, the SDX Session will try to resend the message for the number of times set in this field (typically 2 or 3). Note that an SDX Session configured to continually read inputs and send outputs may not benefit from retries, since input requests and output commands will be repeated anyway.

Timing Parameters

Click on the Timing Tab in the SDX Network Session.

Timing:

Rcv character timeout:
5000 (ms)

Event gap:
0 (ms)

Session gap:
0 (ms)

Probe interval:
0 (sec) ☐ Disable

Connect timeout:
35 (sec)

Session activity timeout:
30 (sec)

Probe Interval - When a remote device quits responding, it can drag down the performance of the entire network waiting for a message response timeout, yet it's important to keep trying to talk to the remote to determine when it comes up again. An SDX Session will use the probe interval to occasionally attempt to communicate with the slave, but not as frequently as if it was up and operational. This way, the slave is checked periodically, but the impact of a "dead" slave on throughput is reduced. To be effective, the Probe Interval Time should be significantly longer than the Response Timeout Time. Note that a value of 0 makes the retries happen at the same rate as the Timeout Time. Recovery is faster this way but the system will timeout and repeat retries every polling cycle slowing the entire system down.

Probe Interval Disable - This will remove probing altogether. In order to reestablish communication with a remote, the associated Network Events must be retrIGGERED. When the remote goes into "com fail", that unit is taken off the polling list if no Network Events have been triggered.

The next two parameters are only used if your SDX is configured for Ethernet or to dial out over a public telephone link for communications. If you're not using an Ethernet or dialup connection, skip these fields.

Connect Timeout (Sec) - After dialing out to establish a link, the modem carrier must be established (dialup) or a TCP/IP connection established (Ethernet) within this time, or SDX will abort the connection.

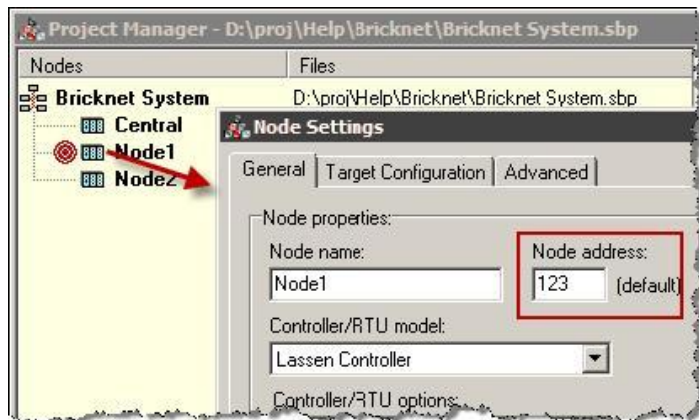
Session Activity Timeout (mS) - If no messaging activity has occurred for this time period, the session will be disconnected (modem hangs up for dialup, or socket disconnects for Ethernet).

For more details, see *Network Sessions Reference* (on page 218) section.

Creating An SDX Destination

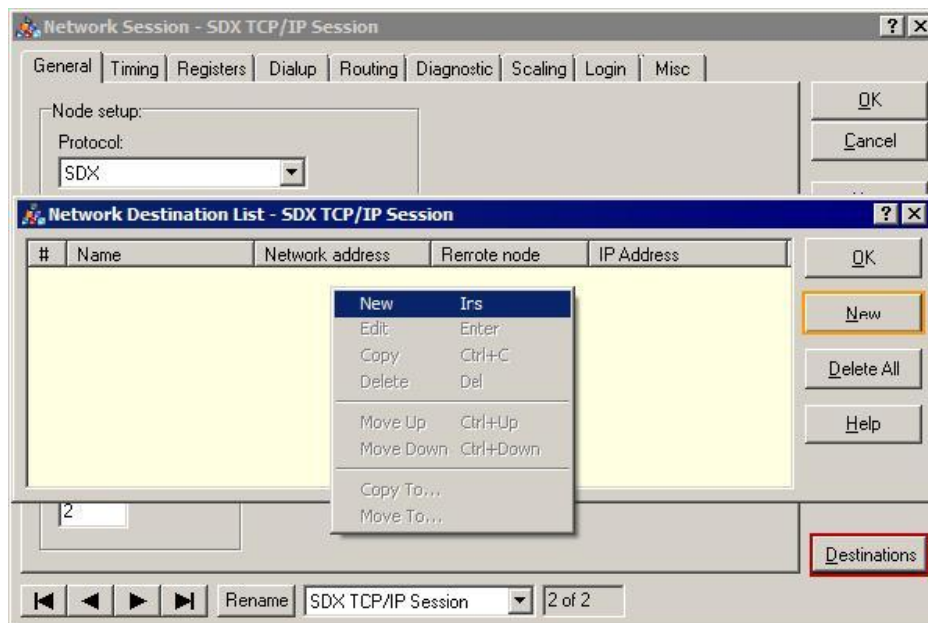


See *The ScadaBuilder Hierarchy* (on page 73).

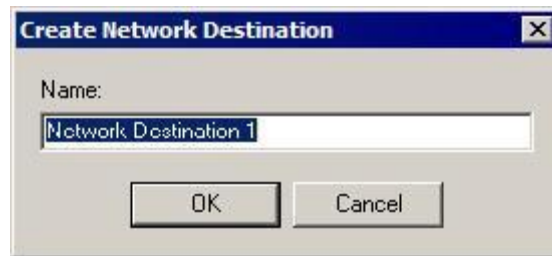


With Controller to Controller communication, to utilize SDX all nodes in an SDX system, all controllers (nodes) should be configured in the current ScadaBuilder Project. Each node should have its default address in the **Node | Settings | General** tab set up so the Destination configuration can get its address easily. If this is not done, the node of interest can still be selected but the address of the remote's SDX Network Session has to be entered manually.

Each remote node that the SDX Network Session needs to talk to must be setup as a Network Destination. To create a Network Destination, Open the SDX Session and click on the Destinations button in the lower right hand corner to bring up the Network Destination List.



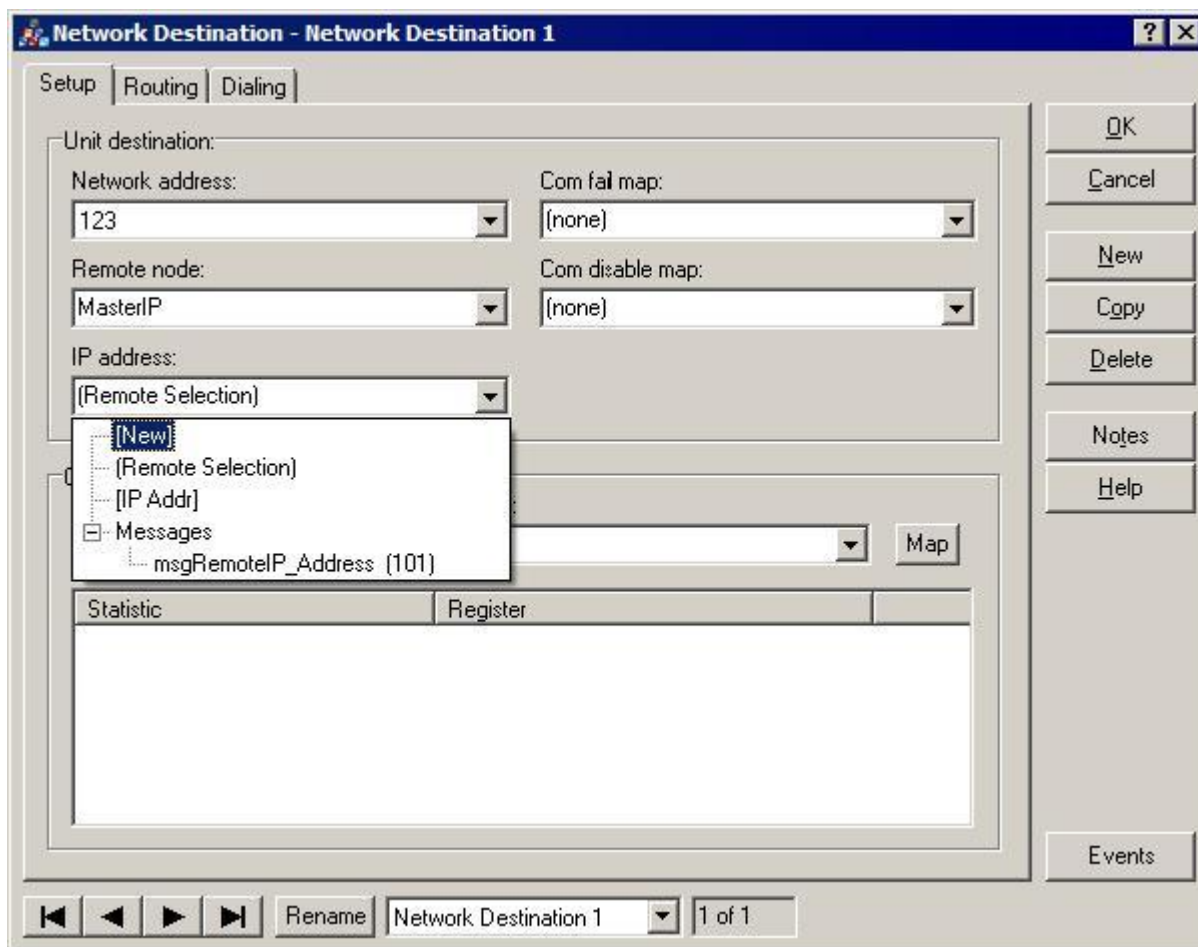
Right click in the list area or click on the New button to start a new Network Destination. You should see this dialog:



Use the default name or give it a new one. You can name it anything you like. If you do use a name it is recommended that the Site name be used and the remote slave address be included so the Destination is readily identified.

Click Ok .

Select the node of interest and if needed enter the Network Address of that unit's SDX Session address or in the case of a remote module the configured address.



In the remote node, you can select another controller in the ScadaBuilder Project or select (Module) in the Remote Node.

IP Address Options For SDX over Ethernet or TCP/IP Interface

Remote Node Option	Supported IP Address Options
(Module)	[IP Addr] -- Hard coded IP address Messages -- Retained Message Register
Node In ScadaBuilder	(Remote Selection) -- IP Address from other Node [IP Addr] -- Hard coded IP address Messages -- Retained Message Register

Network Address Options For SDX serial and STM

Hard coded number

Retained Integer Register

Optionally, you can also map the Com Fail flag to a boolean registers to monitor the general health of the slave. You can also map the Com Disable boolean and communications statistics for more control and monitoring capability. See *Address Com Statistics* (see "Com Statistics" on page 247) for more information.

Creating SDX Network Events



See *The ScadaBuilder Hierarchy* (on page 73).

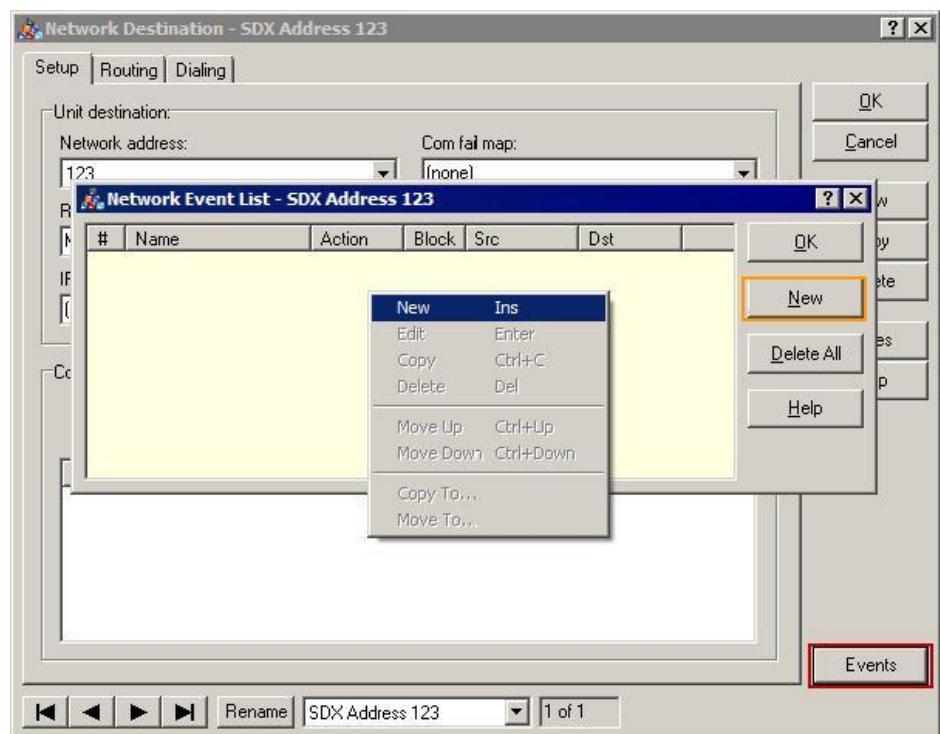
SDX Network Events actually perform two rolls. The first is the traditional Network Event configuration where the SDX Network Event initiates a data transfer to read and write data to other controllers and RTU's.

The second roll is to define where received objects get and place data on the target controller similar to the way Network Message Links work for other protocols.

Let's do the traditional Network Event transaction first.

Traditional Network Events To Read and Write Data

Once the basic Network Session and Network Destination have been set up, you're ready to set up the events for sending messages. Click on the "Events" button in the lower right-hand corner of the Network Destination window. This will bring up a "Network Event List" window.



To create an event, click on “New” or right click in the list area and select New.

Event name:
Event name (optional):
Read UI's

Event message:
Action:
Read

Source: UIReal1 (101) Index: (remote side)

Destination: RemoteAIReal1 (201) Index: (local side)

Block size (or select last register of block): 4 (destination register block)

(Reals)	(Reals)
101 (UIReal1)	201 (RemoteAIReal1)
102 (UIReal2)	202 (RemoteAIReal2)
103 (UIReal3)	203 (RemoteAIReal3)
104 (UIReal4)	204 (RemoteAIReal4)

The resulting Network Event window has an optional field for the event name. Naming your Network Events makes it easier to understand what the events do in the future.

First we need to define the type of data transfer. An event can read registers, write registers, or probe a remote node. Fill in the remote nodes address, and “Action” (read, write or probe) in the appropriate fields.

The Source and Destination registers are read from each node involved and show up in the selection list. There is no other configuration for the register as the data type and existence of the registers is already known from the ScadaBuilder database.

For efficiency, all data transfers are done in blocks of multiple registers. Enter the number of registers to be transferred as the “Block size”.

For a “read” event, data will be transferred from the remote node to matching registers in the local controller. Select the Source data type and Index (register number) in the remote slave, and the FIRST register of the block that the data is to be transferred to in the controller.

For a “write” event, the Source will be the first register of a block of registers in the local controller and the destination will be the corresponding first register in the remote slave that the data will be transferred to. See **Network Events Reference** (on page 255) for more details.

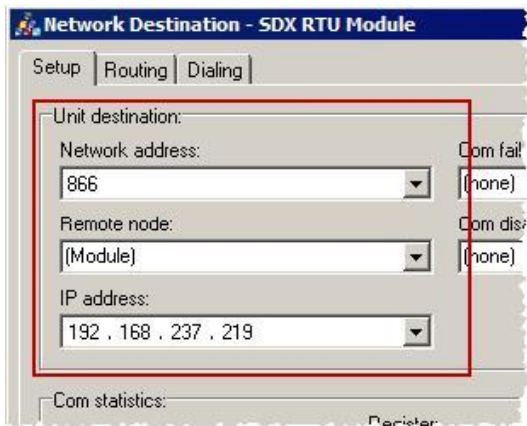
This is a standard Controller to Controller communication over SDX. All register types and indexes are known by the SDX Session according to the Network Destination, Network Event and Registers database in ScadaBuilder.

Click on the Activation tab to setup WHEN this event fires.

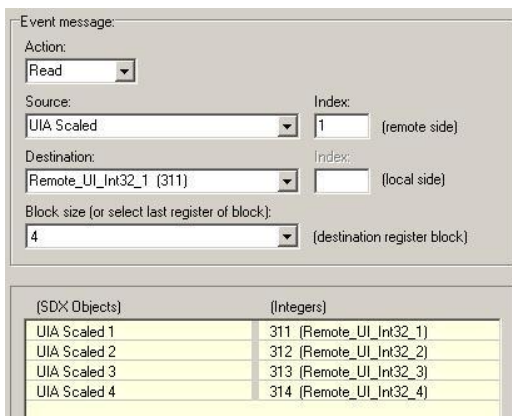
See **Network Event Activation** (on page 259) section for more details.

RTU's and Non Traditional Network Events acting as Network Message Links

If the Network Destination is configured as an IO Module as shown here:

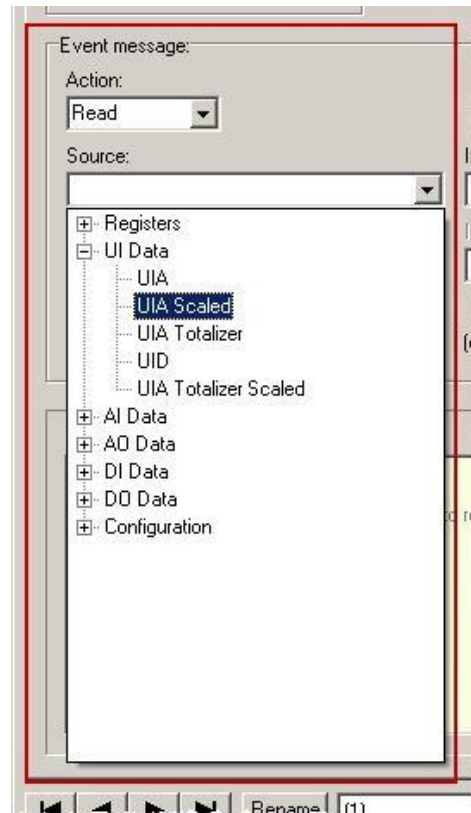


We will configure the Network Event to read a block of 4 UI Scaled registers starting at index 1 (a typical configuration for a Sprite's first 4 UI's).



The Network Event also takes on the roll of a Network Message Link. Let's configure a Network Event for the RTU Network Destination.

If (Module) is selected in the Remote Node field above, a different and more generic list of source registers shows up for a Read Network Event.



The standard Network Event Activation rules apply and we can poll for this data at will if we want to. We can also not configure any Activation at all and let the Sprite simply transfer the data when it sees fit. A Sprite or any Sentry line RTU from ICL can be configured with a Destination IP and a Destination SDX address to send its data on exception or at a configured poll rate.

When the SDX Session receives a message from a remote, it looks to see what SDX address the message is coming from and does a lookup through all the Network Destinations for a matching Network Address. When there is a match (it does not care about the IP address), it then parses the message for all objects then does a lookup on each object in the list of Network Events under the given Destination. When it finds a match, it places the data in the registers on the local controller.

Sprite UI 1-4 -> Sprite Master Mode -> Controller -> Network Session -> Network Destination -> Network Event -> Local Registers 1-4.

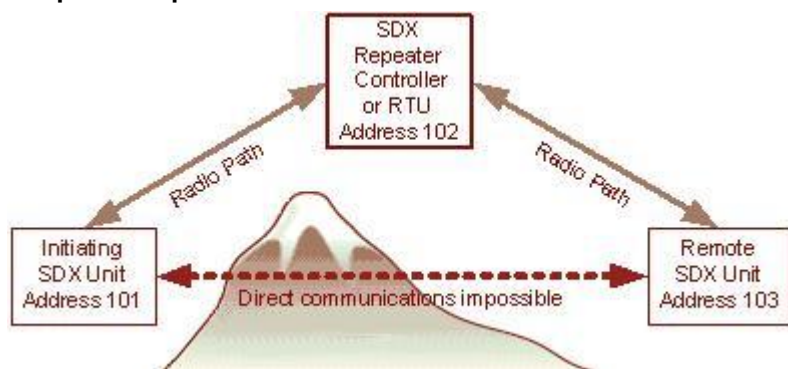
If there is no Network Event matching an object type, then that part of the message is thrown away with no error.

In this way the Network Event acts like a Network Message Link to the data shown in the display area.

Understanding SDX Routing

SDX is an ICL secure protocol used by ICL Controllers and RTU's. By default, each SDX unit using a serial port is a Router or Repeater (meaning it receives route messages and repeats them), commonly referred to as a Store and Forward device. This is extremely useful when using radio systems where a device may not be in range but an interim device can be reached. To use the Routing feature so SDX, Radios must function in a peer to peer configuration where it is possible for all radios in range of each other can hear any transmission. Radios with master/slave relationships cannot be used for this feature. Freewave radios are one device that cannot use this style of Routing--they can only have one master that all radios must be able to hear to function. DIGI (formerly known as MaxStream) radios can use this style of Routing--they are peer to peer radios and can listen in to all radios within range.

Simple Example



The Routing engine and the information necessary is built into the protocol and transmitted with each data packet. Because of this built-in functionality, Routing configurations only need to be done on units that initiate messages over the network.

In the above situation, the initiating unit 101 has a Network Destination for address 103. That Destination is programmed with one hop -- 102.

Essentially, this configuration at the initiating unit inserts a "hop" of 102 into the transport layer of the SDX message packet sent over the radio link.

The message is transmitted from the initiating unit 101 and the 102 unit sees itself in the hop table then re-transmits the data to the destination 103. A hop count is also included in the message so that all units know how many hops are necessary get to the destination address. In this case it is only one hop.

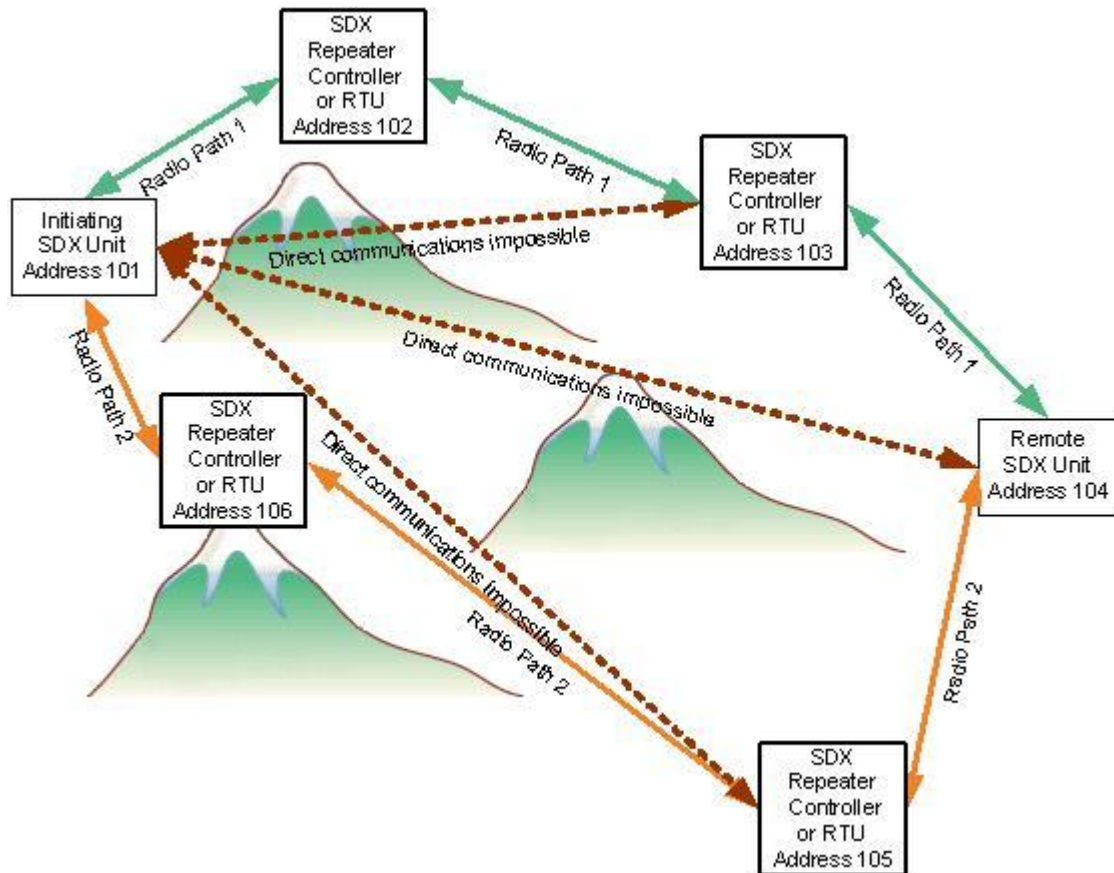
The 103 unit receives its message, sees the hop table in the message and responds with the proper 102 hop in the message.

102 sees that it is a router again and repeats the message back to unit 101.

The transaction is complete.

Multiple Path Example

Multiple routes may also be supported for up to 10 hops and allows for much more complicated systems where mountainous areas make for difficult radio path situations.



In this configuration there is a choice to make. The goal is to get a message from the initiating unit 101 to the remote 104. There are two possible paths to use to get the data.

Path 1 (in green) hops from 101 to 102 to 103 and finally ends up at 104.

Path 2 (in orange) hops from 101 to 106 to 105 and finally ends up at 104.

While both paths take the same number of hops (repeating twice each direction for a total of 6 messages), Path 1 is probably the better choice as it has a more consistent distance between hops where as Path 2 requires a long distance transmissions between 106 and 105.

Both Path 1 and Path 2 can be programmed in the initiating unit 101 in the SDX Network Destinations.

Path 1 is the primary and enabled by default. Path 2 can be disabled by default. Upon a failure in Path 1 (say address 103 stop communicating because of power or radio failure), Path 2 can then be enabled to take over. Though it is not an optimal situation, if Path 2 is functional, then the data gets where it needs to go.

Setting Up An SDX Routing Example

Routing is setup only in an initiating unit of the system. Since the protocol is peer-to-peer, any unit may be setup with routing information in the Network Destination. Here we will setup the multiple path example above and configure two Destinations--one for Radio Path 1 and one for Radio Path 2. Both Destinations will go to address 104.

Create a serial Network Port | Network Session (with address 101). See the previous sections in *Secure Data Exchange (SDX and STM) Protocols* (on page 287) for more details on creating these records.

Network Session - SDX Session Addr 101

General Timing Registers Dialup Routing D

Timing:

Rcv character timeout:
5000 (ms)

Event gap:
0 (ms)

Session gap:
0 (ms)

Probe interval:
60 (sec) ☐ Disable

Connect timeout:

In the SDX Network Session we are going to configure a Probe Time which will allow the system to go into a slow poll mode should any Network Destination go into communications failure. See *Probe Interval* (on page 225) for details on what this does.

We will create two Network Destinations--both with a Network Address of 104. See *Creating An SDX Destination* (on page 291) for details. We need to map two communications flags for each Destination record:

For Path 1, we need to map the ComFail to a boolean. For our exercise here we do not need to map the Com Disable but good practice is to do it anyway.

Network Destination - Radio Path 1 Addr 104

Setup Routing Dialing

Unit destination:

Network address:
104

Remote node:
(Module)

IP address:

Com fail map:
RadioPath1ComFail (1001)

Com disable map:
RadioPath1ComDisable (1002)

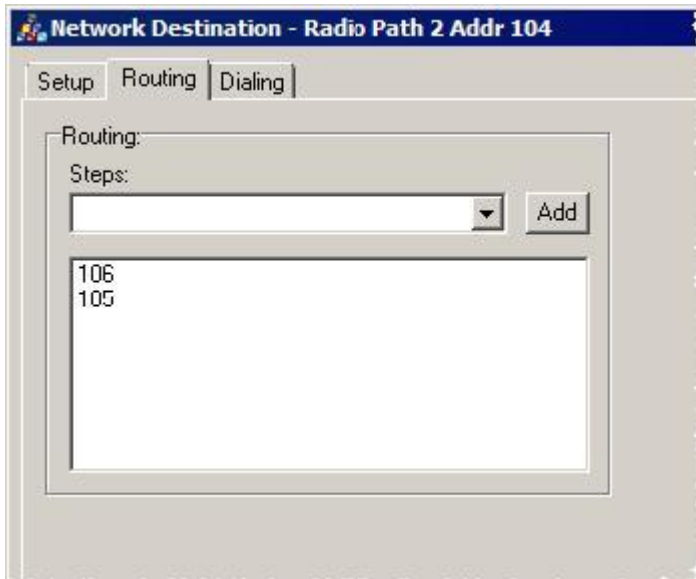
For Path 1 we simply hard code the routing entries.



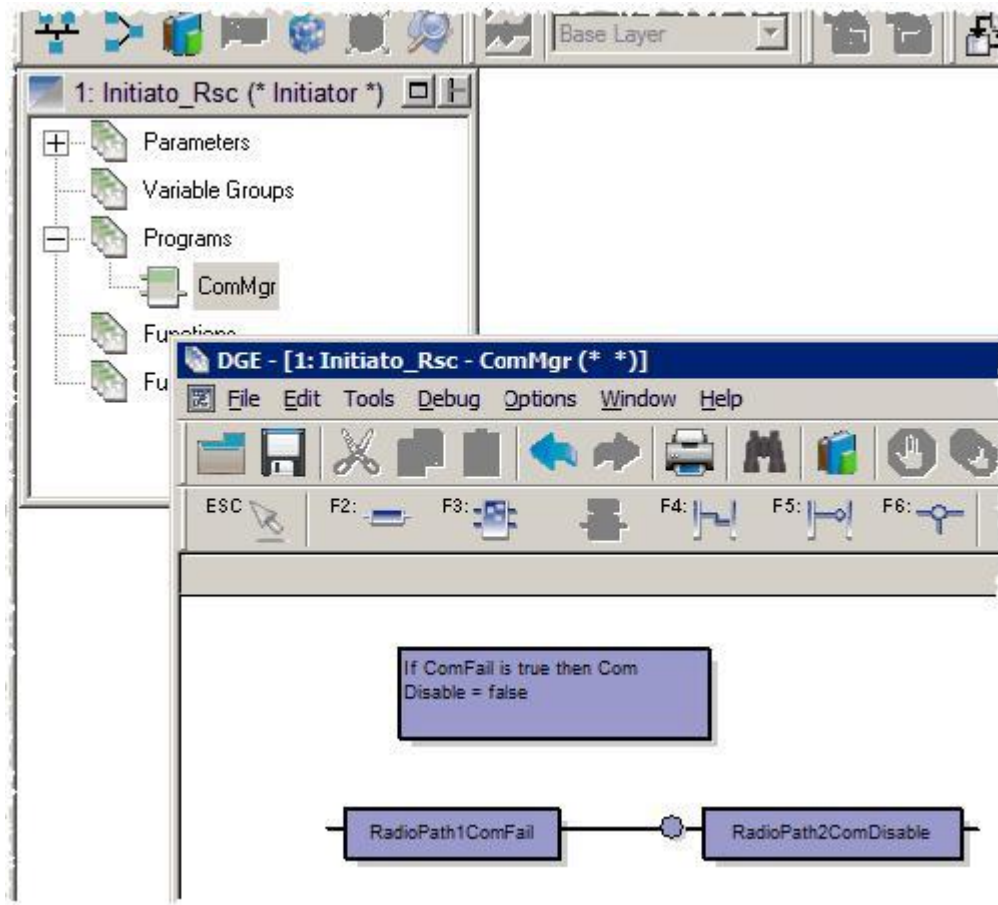
For Path 2, we need to map the Com Disable:



For Path 2 we simply hardcode the routing entries:



From here we just need to do a little ISaGRAF logic to tie the two flags together:



When the RadioPath1ComFail boolean goes true, then the RadioPath2ComDisable boolean will be set to FALSE enabling the Path 2 Destination. When communication over Path 1 is restored (in our configuration here it will try once every 60 seconds while Path 1 is in communications failure) Path 2 will be disabled.

The only thing you need to make sure of now is that the two Network Destinations have the same Network Event configuration.

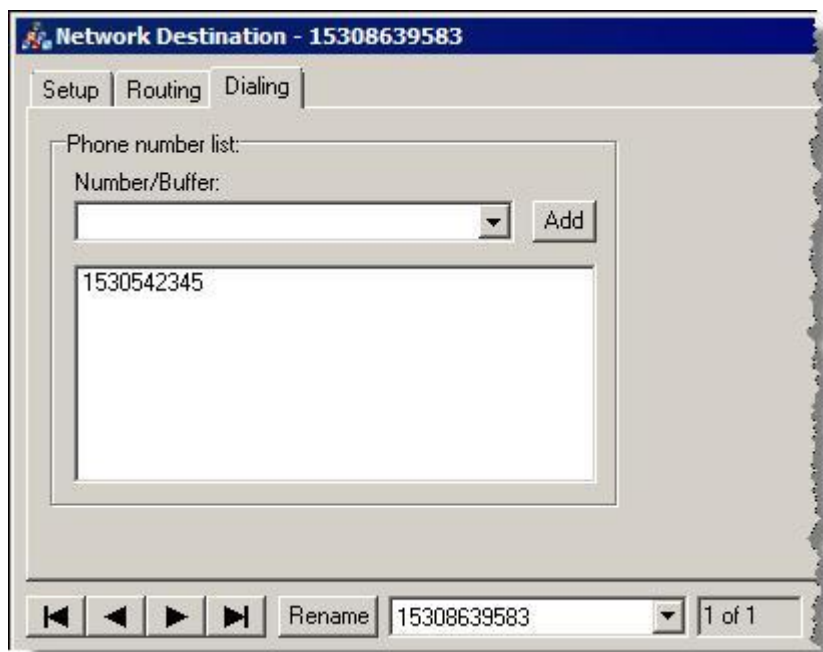
Setting Up An STM Interface

STM stands for SDX Text Messaging. An STM Interface works much like an SDX serial interface with only some basic differences.

- STM must use a **cellular modem** (usually an ICL Messenger Series Cellular Modem) over either a serial or USB port. See *Using a Cellular Modem Network Port* (on page 210) for more details.
- Time synchronization is crucial to the proper operation as old messages must be rejected if there is an interruption in coverage or controller operations (such as being powered off for a period of time). The Controller's clock may be synchronized from the Cell Modem.
- The Time Zone and Day Light Savings time flag should be properly configured. See *Mappings* (see "Mappings Reference" on page 493) section for more details.
- STM (not SDX) protocol must be selected in the Network Session.

Setting up the Network Destination

The only configuration that differs with an STM Network Destination (as compared to an SDX Network Destination) is the phone number.



Always include the 1 and the area code of the phone number.

All other configurations are the same as the SDX Protocol.

SDX Routing is not supported over cellular links.

DF1 Protocol



See *The ScadaBuilder Hierarchy* (on page 73).

Due the installed base of Allen-Bradley programmable logic controllers and the high cost of protocol adapter modules, the DF1 protocol is a popular means of interfacing these PLCs with ICL controllers. In the DF1 protocol, a single Master communicates with up to 254 slaves. Slaves do not send messages on their own; they respond to messages from the Master. DF1 is designed to operate over serial networks; RS-232 for short point-to-point connections, RS-485 for longer distance hard-wired networks, and radios and modems for even longer distances. DF1 can support three types of data; bits, integers and floating point numbers. DF1 over Ethernet is not supported at this time.



ICL controllers support the serial, half-duplex version of DF1, both BCC and CRC error checking. BCC/CRC selection is in the Misc Tab of the Network Session.

Troubleshooting



DF1 Half Duplex constantly talks over a link layer on a half duplex serial port. The link layer must be working for any register transaction to be successful. The port lights will show constant activity when the baud rate, half duplex (on the other controller or RTU), error checking and address are configured correctly between two or more devices.

If there is not constant activity, check all of these parameters first. One of them is likely the problem

Creating a DF1 Master Interface

A DF1 Master requires more effort to configure than a DF1 Slave because the Master must initiate the messages that request data from the Slaves or send data to the Slaves. Slaves simply respond to data requests.

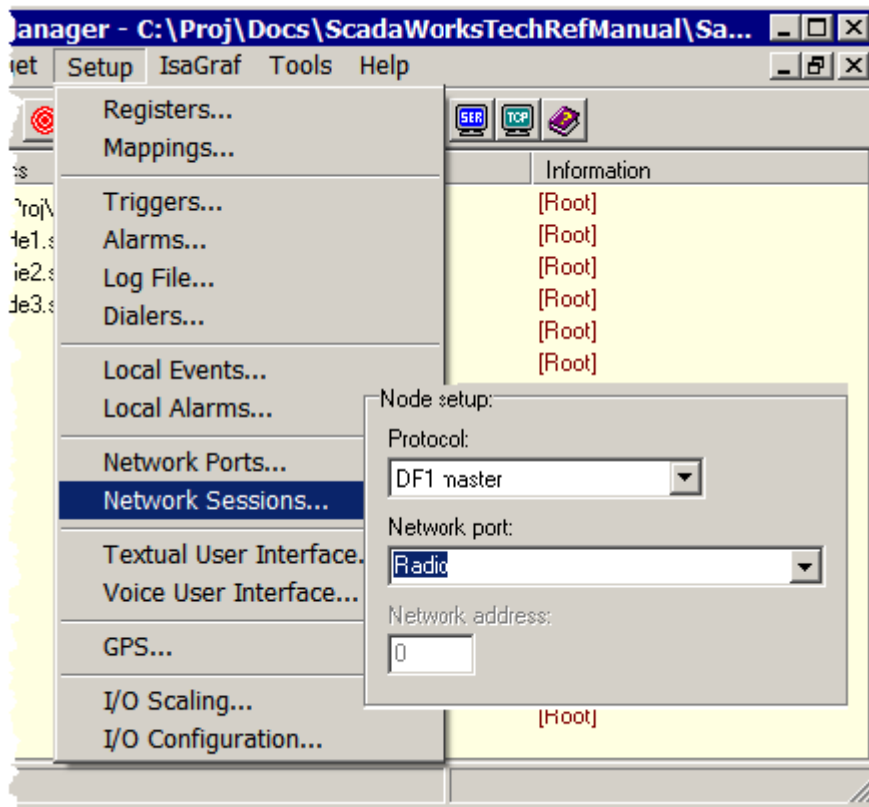
Some thought should go into how you want to set up your DF1 Master. In its simplest form, the Master can continuously request data from DF1 Slaves, and continuously write data to them. This is the easiest form of Master to create, but it is also the least efficient.



With a little more effort, it's possible to prioritize the frequency of the events that cause messages to be sent and use Triggers to send data intelligently when it changes. See Using Triggers for more details.

For example, remote outputs can be updated when outputs need to be changed instead of constantly writing the same output data over and over. This is accomplished by defining Triggers. A single Trigger can look for a change in a block of registers. When a change occurs, the Trigger can cause a DF1 message to be sent that updates the remote outputs. Triggers can be added at any time as needed.

You will need to set up a Network Port that defines the basic hardware level communications elements such as baud rate, parity and hardware interface (RS-232, RS-485, etc.) or you can simply use the defaults for now and change the Network Port to your liking later on.



To create a DF1 Master session, select the **Setup | Network Sessions...** menu. A dialogue window will pop up for naming the session. Accept the default name or enter a new one. If you already have at least one Network Session, you can also simply click on the “New” button on the right-hand side of the Network Session window.

Select the protocol as DF1 Master.

Select your Network Port.

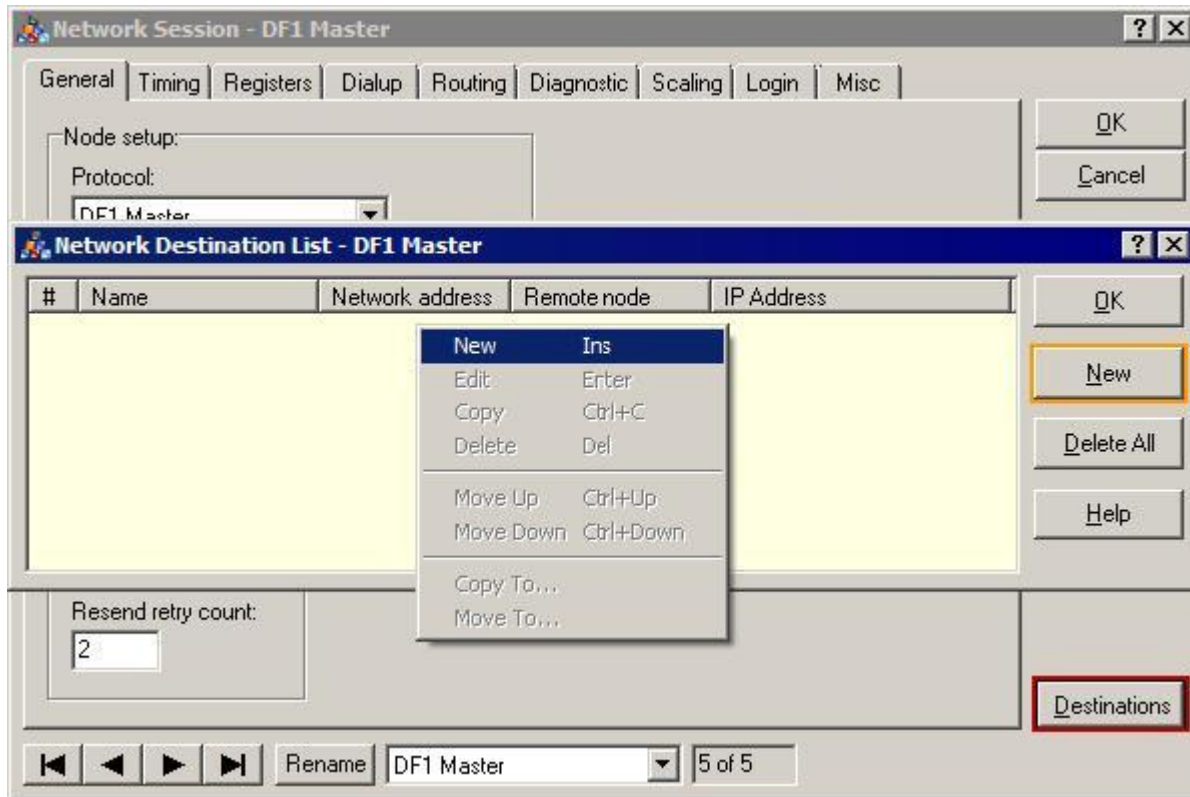
For more details, see *Network Sessions Reference* (on page 218) section.

Creating a DF1 Master Destination

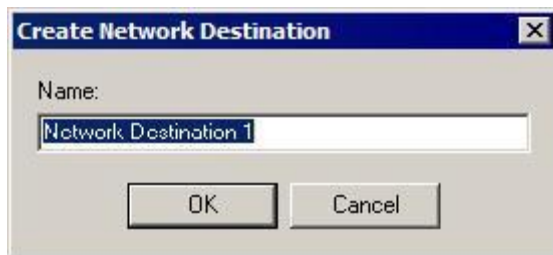


See *The ScadaBuilder Hierarchy* (on page 73).

Each remote slave that the DF1 Master Network Session needs to talk to must be setup as a Network Destination. To create a Network Destination, Open the DF1 Master Session and click on the Destinations button in the lower right hand corner to bring up the Network Destination List.



Right click in the list area or click on the New button to start a new Network Destination. You should see this dialog:



Use the default name or give it a new one. You can name it anything you like. If you do use a name it is recommended that the Site name be used and the DF1 Slave address be included so the Destination is readily identified.

Click Ok .

Enter the Network Address.

Network Destination - Network Destination 1

Setup | Routing | Dialing

Unit destination:

Network address: 1

Com fail map: (none)

Remote node:

Com disable map: (none)

IP address:

Com statistics:

Select Register Map

Statistic	Register

OK Cancel New Copy Delete Notes Help Events

Navigation: [Back] [Previous] [Next] [Forward] Rename Network Destination 1 1 of 1

Optionally, you can also map the Com Fail flag to a boolean registers to monitor the general health of the slave. You can also map the Com Disable boolean and communications statistics for more control and monitoring capability. See *Address Com Statistics* (see "Com Statistics" on page 247) for more information.

Creating DF1 Master Events



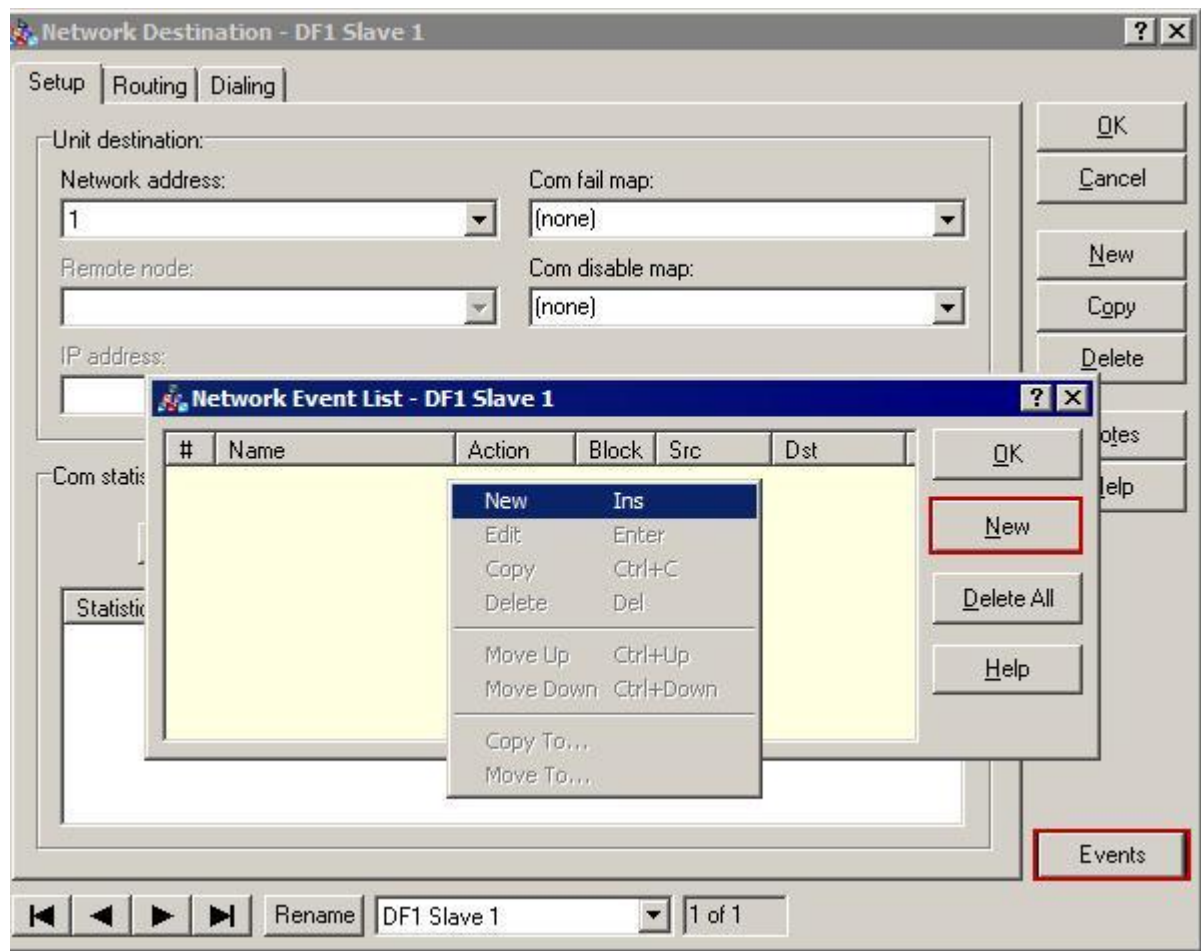
See *The ScadaBuilder Hierarchy* (on page 73).

Once the basic Network Session and Network Destination have been set up, you're ready to set up the Network Events for your Master.

Click on the "Events" button in the lower right-hand corner of the window to bring up a "Network Event List" window.

To create an event, click on "New" or right click in the list and select New.

The resulting Network Event window has an optional field for the event name. Naming your Network Events makes it easier to understand what the events do in the future.



Event Activation Options

Event name:
Event name (optional):
Read DF1 AI's

Event message:
Action:
Read

Source: N7 (integer) Element: 1 (remote side)

Destination: REM_AI1 (601) Element: (local side)

Block size (or select last register of block): 20 (destination register block)

(DF1 References)	(Booleans)
File: N7, Elem: 1	601 (REM_AI1)
File: N7, Elem: 2	602 (REM_AI2)
File: N7, Elem: 3	603 (REM_AI3)
File: N7, Elem: 4	604 (REM_AI4)
File: N7, Elem: 5	605 (REM_AI5)
File: N7, Elem: 6	606 (REM_AI6)
File: N7, Elem: 7	607 (REM_AI7)
File: N7, Elem: 8	608 (REM_AI8)
File: N7, Elem: 9	609 (REM_AI9)
File: N7, Elem: 10	610 (REM_AI10)

First we need to define the type of data transfer. An event can read registers, write registers, or probe a slave. Fill in the remote slave address to exchange data with and select an action (read, write or probe) in the appropriate fields.

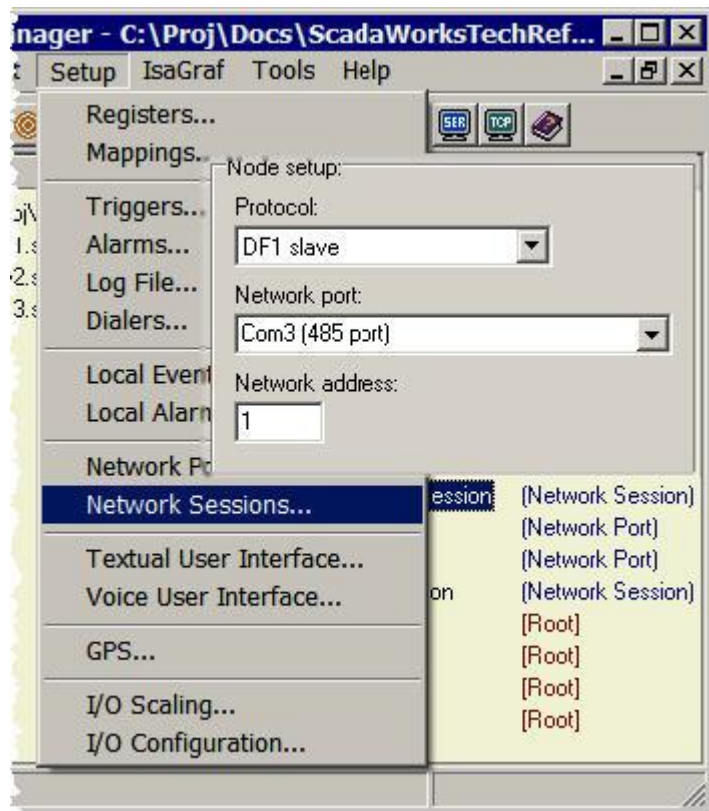
For efficiency, all data transfers are done in blocks of multiple registers. Enter the number of registers to be transferred as the "Block size".

For a "read" event, data will be transferred from the slave's B3, N7 or F8 registers (or you can enter your own file number) to matching registers in the controller. Select the Source data type and Index (register number) in the remote slave, and the FIRST register of the block that the data is to be transferred to in the controller.

For a "write" event, the Source will be the first register of a block of registers in the controller and the destination will be the corresponding first register in the remote slave that the data will be transferred to.

Click on the Activation tab. See *Network Event Activation* (on page 259) section for more details.

Creating a DF1 Slave Interface



A DF1 Slave session is easier to configure than a DF1 Master because a Slave simply needs to listen and respond to requests and commands from the Master. There's no event configuration!

To create a DF1 Slave session, select the **Setup | Network Sessions...** menu. A dialogue window will pop up for naming the session. Accept the default name or enter a new name. If you already have at least one Network Session already, you can also simply click on the "New" button on the right-hand side of the Network Session window.

Protocol - Once you have a new Network Session window, select the protocol; DF1 Slave.

Network Port - After selecting the protocol, choose the Network Port. ScadaBuilder only displays the port names of ports where the configuration is compatible with the selected protocol. If you don't see the port that you want named as a choice, go back and check the Network Port configuration.

Network Address - A DF1 Slave must be assigned an address between 1 and 255. You also have the option of using DIP switches on the Controller (EtherLogic or ScadaFlex Plus family) or a register for changing the slave address. This is accomplished through fields under the "Misc" Tab. If "Register" is chosen, you must select the tag name of an integer register and configure it as "retained" (nonvolatile).



In order to use the DF1 slave session, we must first assign file types to register ranges using Network Message Links.

Network Message Link - Network Msg Link 1

Message link:

Start register: AI1 (4030) Data pack: (default)

Block size (or select last register of block): 4

File number: N7 (integer) Element: 1

Link options:

Register mode: (default) Integer cast type: signed Access mode: any

(Integers)	Msg Registers
4030 (AI1)	File: N7, Elem: 1
4031 (AI2)	File: N7, Elem: 2
4032 (AI3)	File: N7, Elem: 3
4033 (AI4)	File: N7, Elem: 4

Reg count: 4 Byte count: 16

Display

Navigation: [Previous] [Next] [First] [Last] [Rename] Network Msg Link 1 1 of 1

Shown here is a Network Message Link that exposes AI1 through AI4 to N7:1 (Element 1) through N7:4. Notice that the Element numbers have nothing to do with the internal numbering of the controller registers.

Any file number may be used (0 to 255). Shown here are the pre-defined file numbers. If you would like to use a non-standard file number, simply select number and type it into the source or destination register's file number field. See

Network Message Link Reference (on page 274) for more information.

DNP 3 Protocol



See *The ScadaBuilder Hierarchy* (on page 73).

DNP 3.0 is often used in the power industry as a high reliability network. DNP 3 operates in both Master and Slave modes. It supports running over RS-232 or RS-485 and TCP/IP connections. It does not support dialup connections.

For a full report on Pinnacle series and later controller DNP 3 compliance, go to the web site here: ***DNP 3 Compliance Document*** http://www.iclinks.com/public_ftp/TechSupport/DNPPProfile/PinnacleDnpProfile.htm.

Master Side Supported Data Types

Allows the node originate register based commands in the DNP3 protocol.

Here is a table of supported commands:

Group	Message Type (Data Type) Format
Binary Inputs	Binary Pack
	Binary Status
Binary Outputs	Binary Pack
	Binary Status
Binary Counters	Integer 32 bit
	Integer 16 bit
Analog Inputs	Integer 32 bit
	Integer 16 bit
	Floating point
Analog Outputs	Integer 32 bit
	Integer 16 bit
	Floating point
Class	Class 0 static date and Class 1-3 Variant Data Types

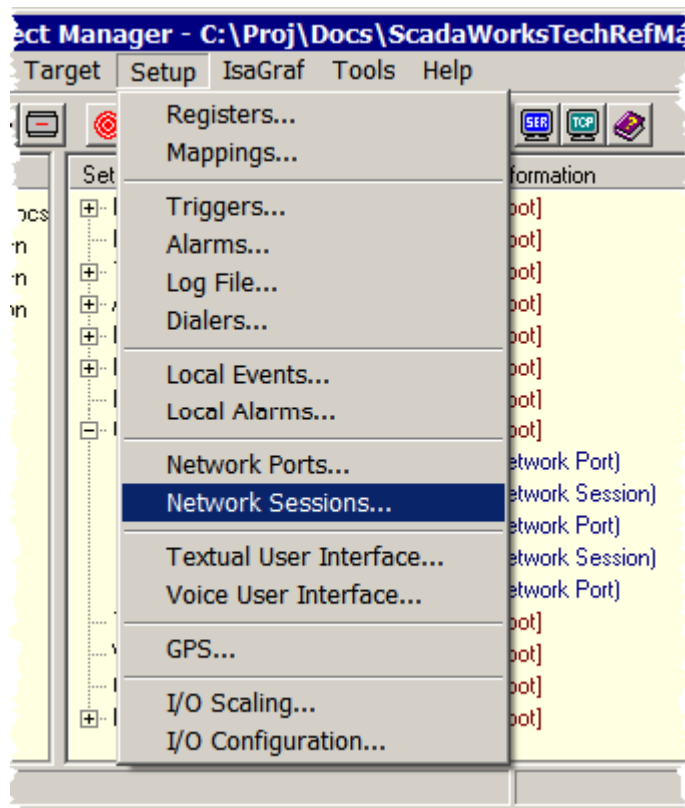
Special DNP 3 Diagnostic Statistics

Below the standard application layer where register data is transferred, DNP 3 has a link layer as well as a confirmation layer to the protocol resulting in a robust link but also increasing complexity when troubleshooting. The following table illustrates the extra network statistic available to aid in the troubleshooting process:

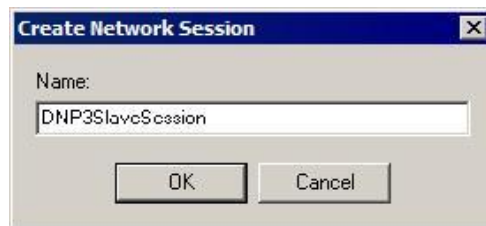
DNP3 Statistics	Master	Slave
Confirm Timeout	√	√
Confirm Receive	√	√
Confirm Transmit	√	√

Local Error code	√	√
Remote Error code	√	
Indicator Bits	√	
DNP3 Link Layer Statistics		
Transmit Primary	√	√
Receive Primary	√	√
Transmit Secondary	√	√
Receive Secondary	√	√
Confirm Timeout	√	√
CRC Error	√	√
Fragment Overflow	√	√
Link Reset	√	√
Resp ACK	√	√
Resp NAK	√	√
Resp Busy	√	√
Resp Ready	√	√
Resp Service Error	√	√

Creating a DNP 3 Slave Interface



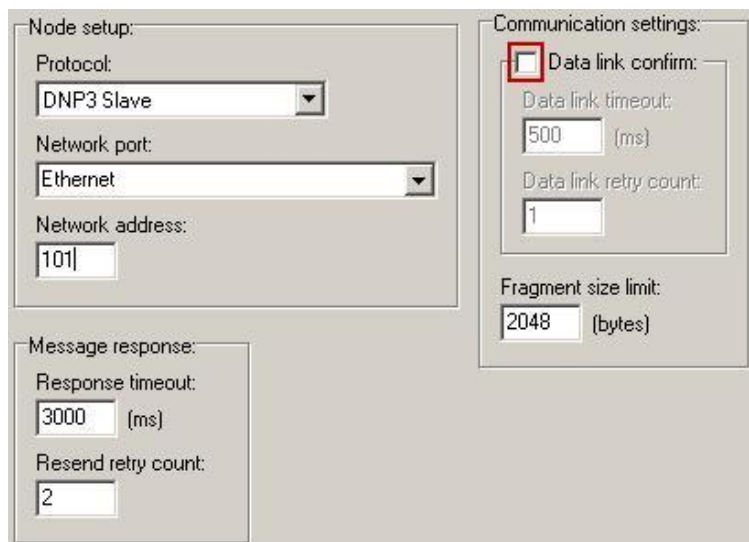
A DNP 3 Slave session is easier to configure than a DNP 3 Master because a Slave simply needs to listen and respond to requests and commands from the Master. To create a DNP 3 Slave session, select the **Setup | Network Sessions...** menu. A dialogue window will pop up for naming the session. Accept the default name or enter a new name. If you already have at least one Network Session already, you can also simply click on the “New” button on the right-hand side of the Network Session window.



Protocol - Once you have a new Network Session window, select the protocol; DNP 3 Slave.

Network Port - After selecting the protocol, choose the Network Port. ScadaBuilder only displays the port names of ports where the configuration is compatible with the selected protocol. If you don't see the port that you want named as a choice, go back and check the Network Port configuration.

Network Address - A DNP 3 Slave must be assigned an address between 1-65519.



There are several more layers to timeout, comfail, and recovery in the DNP 3 protocol. The data link timeout is the lowest layer (how long to we wait for a confirmation of a message). You can choose to use the Data Link Confirm option or not. It must be the same for both Master and Slave (the same checkbox is in a DNP 3 Master session if you are using an ICL controller to read and write data to this DNP 3 Slave).

Configuring the Master Address for the Slave

A DNP 3 Slave must also be configured to know what Master address to talk with in the DNP tab of the DNP Slave Network Session.

General Timing Registers Dialup Routing Diagnostic Scaling Login Misc DNP

Unsolicited responses:

☒ Enable Class 1: Hold time: 60 (sec) Hold count: 10

☐ Enable Class 2: Hold time: 60 (sec) Hold count: 10

☐ Enable Class 3: Hold time: 60 (sec) Hold count: 10

☐ Send initial unsolicited response at startup ... ☒ Include static data

Outstation settings:

Event buffer size: 1024

Master address: 100

Master IP address: 192.168.237.212

Operate timeout: 10 (sec) Time sync interval: 5 (min) Response confirmation: Event data & multi-fragment responses.

Required for Serial and TCP/IP Network Ports

Required for TCP/IP Network Ports

For Serial Network Ports only the Master Address is required. For TCP/IP Network Ports, both the Master Address and the Master IP Address are required. For the IP address, you can specify an Node within the project, a Retained Message register or a static IP address.

Master IP address:

msgSlaveIPAddress (1)

[New]

[IP Addr]

Nodes

MasterIP: Eth1

SlaveIP: Eth1

Messages

msgSlaveIPAddress (1)

Msg Links



In order to use the DNP 3 slave session, we must first assign group types to register ranges using Network Message Links.

See **Network Message Link Reference** (on page 274) for more information.

Message link:

Start register:
UIReal1 (101)

Block size (or select last register of block):
4

Group/variation:
Analog Inputs (32-bit no flags)

Index:
1

Event data:

Event class:
(none)

Threshold (input deadband):
1

☐ Report time with event data.

Local Registers: (Reals)	Msg Registers
101 (UIReal1)	Analog Inputs 1
102 (UIReal2)	Analog Inputs 2
103 (UIReal3)	Analog Inputs 3
104 (UIReal4)	Analog Inputs 4

Reg count: 4 Byte count: 16

Display

Any Network Message Link created can be assigned to a DNP 3 class (1 through 3). Just select the Data Class to apply this Message Link to. When a DNP 3 master asks for a class of data, the controller will respond with the data in every Message Link assigned to that class.

With the Class set to "None", the DNP 3 Slave will only respond with this data if asked for it as a static request for Analog Inputs 1 through 4 (reals).

Shown here is a Network Message Link that exposes UIReal1 through UIReal4 to Analog input (message type) index 1 through 4. Notice that the Analog input numbers have nothing to do with the internal numbering of the controller registers.

When the system receives a request for floating point Analog Input between 1 and 4 on the index, then it will respond with the "static" or live value of the register.

Any supported data type may be used:

- Binary Inputs (packed)
- Binary Inputs (status)
- Binary Outputs (packed)
- Binary Outputs (status)
- Binary Counters (32-bit w/ flags)
- Binary Counters (16-bit w/ flags)
- Binary Counters (32-bit no flags)
- Binary Counters (16-bit no flags)
- Analog Inputs (32-bit w/ flags)
- Analog Inputs (16-bit w/ flags)
- Analog Inputs (32-bit no flags)
- Analog Inputs (16-bit no flags)
- Analog Inputs (float w/ flags)
- Analog Outputs (32-bit w/ flags)
- Analog Outputs (16-bit w/ flags)
- Analog Outputs (float w/ flags)

Creating a DNP 3 Master Interface

A DNP 3 Master requires more effort to configure than a DNP 3 Slave because the Master must initiate the messages that request data from the Slaves or send data to the Slaves. Slaves simply respond to data requests.

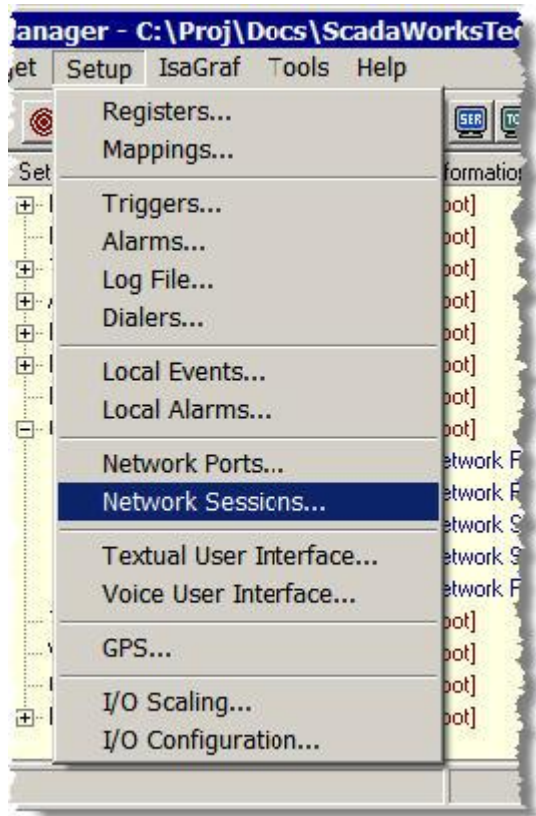
Some thought should go into how you want to set up your DNP 3 Master. In its simplest form, the Master can continuously request data from DNP 3 Slaves, and continuously write data to them. This is the easiest form of Master to create, but it is also the least efficient.



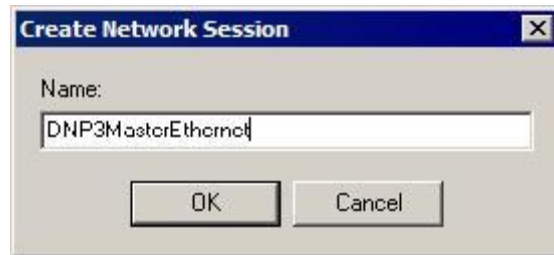
With a little more effort, it's possible to prioritize the frequency of the events that cause messages to be sent and use Triggers to send data intelligently when it changes. See Using Triggers for more details.

For example, remote outputs can be updated when outputs need to be changed instead of constantly writing the same output data over and over. This is accomplished by defining Triggers. A single Trigger can look for a change in a block of registers. When a change occurs, the Trigger can cause a DNP 3 message to be sent that updates the remote outputs. Triggers can be added at any time as needed.

You will need to set up a Network Port that defines the basic hardware level communications elements such as baud rate, parity and hardware interface (RS-232, RS-485, Ethernet, etc.) or you can simply use the defaults for now and change the Network Port to your liking later on.



To create a DNP 3 Master Session, select the **Setup | Network Sessions...** menu. A dialogue window will pop up for naming the session.



Accept the default name or enter a new one. If you already have at least one Network Session, you can also simply click on the “New” button on the right-hand side of the Network Session window.

Select the protocol as DNP3 Master.

Select your Network Port: Ethernet or Serial.

A DNP 3 Master must be assigned an address between 1 and 65519.

There are several more layers to timeout, comfail, and recovery in the DNP 3 protocol.

The data link timeout is the lowest layer and has its own set of Network Statistics. Depending on the configuration of the DNP 3 slave you are talking to, you can choose this option. Both sides must agree for the Data Link layer and the application layer to both work.

Enabling the Data Link layer will increase the data traffic across the link.

The screenshot shows a configuration window with three main sections:

- Node setup:**
 - Protocol: DNP3 Master (dropdown)
 - Network port: Ethernet (dropdown)
 - Network address: 100 (text box)
- Communication settings:**
 - ☒ Data link confirm: (checkbox, highlighted with a red box)
 - Data link timeout: 500 (ms) (text box)
 - Data link retry count: 1 (text box)
 - Fragment size limit: 2048 (bytes) (text box)
- Message response:**
 - Response timeout: 3000 (ms) (text box)
 - Resend retry count: 2 (text box)

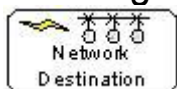
The Fragment size limit shown can reduce the maximum packet size sent over the link. This is useful for modems and radio modems that have a limited buffer size. The messages will be broken up into manageable sizes based on this parameter.



Any of the layers timing out will cause a message to fail. Statistics are provided in the **Network Session | Diagnostic** tab and in the Network Destination Statistics for troubleshooting these extra layers.

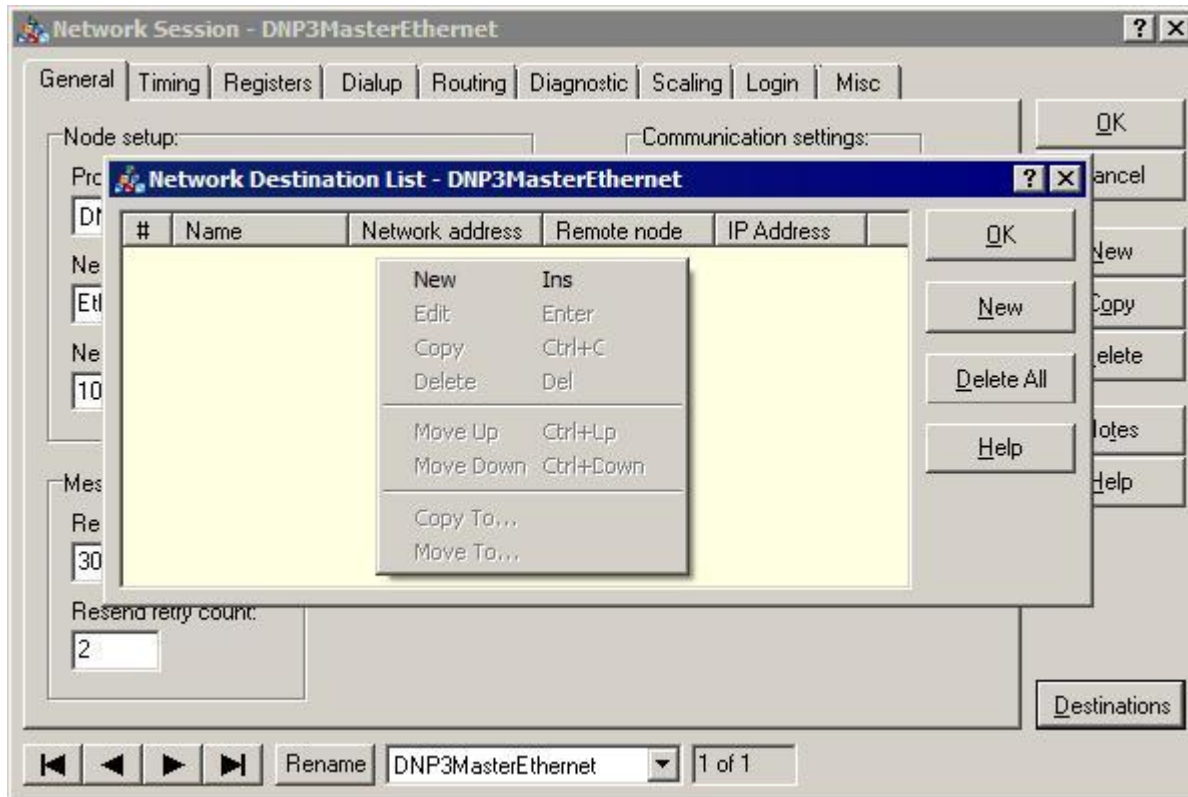
For more details, see *Network Sessions Reference* (on page 218) section.

Creating a DNP 3 Master Destination

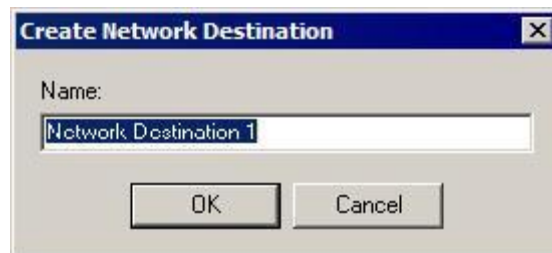


See *The ScadaBuilder Hierarchy* (on page 73).

Each remote slave that the DNP 3 Master Network Session needs to talk to must be setup as a Network Destination. To create a Network Destination, Open the DNP 3 Master Session and click on the Destinations button in the lower right hand corner to bring up the Network Destination List.



Right click in the list area or click on the New button to start a new Network Destination. You should see this dialog:



Use the default name or give it a new one. You can name it anything you like. If you do use a name it is recommended that the Site name be used and the DNP 3 Slave address be included so the Destination is readily identified.

Click Ok .

Setup | Routing | Dialing

Unit destination:

Network address: 101 Com fail map: (none)

Remote node: (none) Com disable map: (none)

IP address:

[New]
[IP Addr]
Nodes
MasterIP : Eth1
SlaveIP : Eth1

Map

Statistic Register

Enter the Network Address.

For a DNP 3 TCP/IP Destination, an IP address or Node must be selected.

Optionally, you can also map the Com Fail flag to a boolean registers to monitor the general health of the slave. You can also map the Com Disable boolean and communications statistics for more control and monitoring capability. See *Address Com Statistics* (see "Com Statistics" on page 247) for more information.

DNP 3 statistics are especially useful for troubleshooting sub layer issues that might come up with a Non-ICL DNP 3 master. The expanded list of statistics are:

Each of these statistics can be mapped to a 32 bit Integer and displayed on a TUI or other HMI for troubleshooting live communications.

- ☒ transmit command
- ☒ receive response
- ☒ receive command
- ☒ transmit response
- ☒ receive timeout
- ☒ bad content
- ☒ configuration error
- ☒ success percent
- ☒ last rcv'd command
- ☒ confirm timeout
- ☒ confirm receive
- ☒ confirm transmit
- ☒ local error code
- ☒ remote error code
- ☒ indicator bits
- ☒ (Link) transmit primary
- ☒ (Link) receive primary
- ☒ (Link) transmit secondary
- ☒ (Link) receive secondary
- ☒ (Link) confirm timeout
- ☒ (Link) CRC error
- ☒ (Link) fragment overflow
- ☒ (Link) link reset
- ☒ (Link) resp ACK
- ☒ (Link) resp NAK
- ☒ (Link) resp busy
- ☒ (Link) resp ready
- ☒ (Link) resp service err

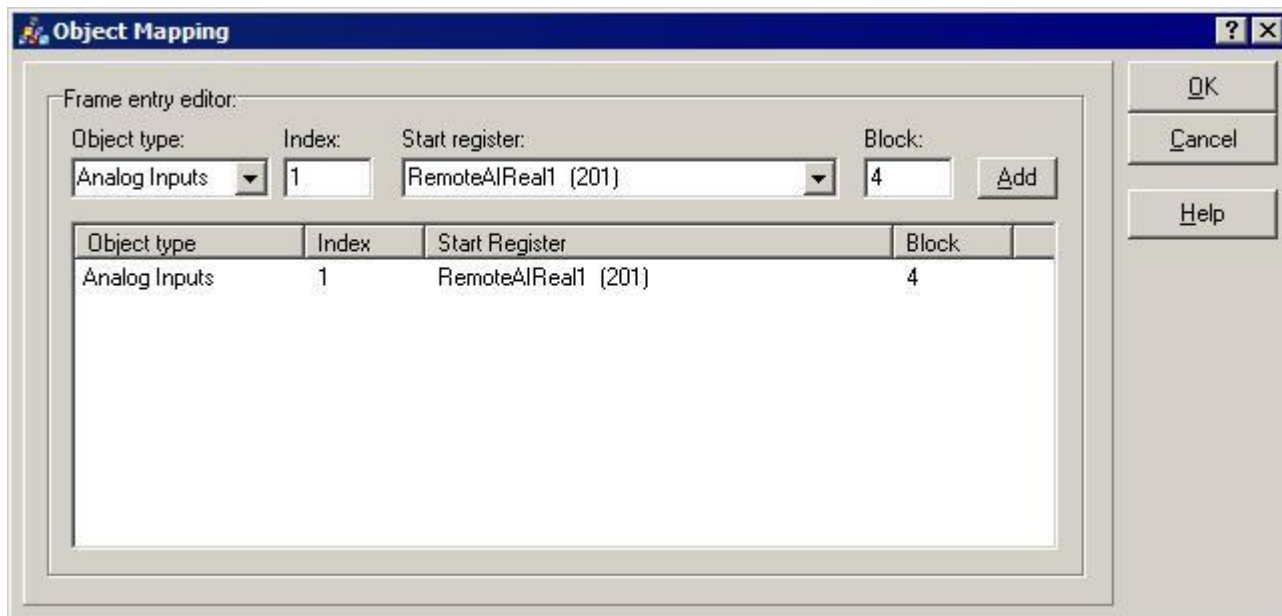
Creating DNP 3 Master Objects



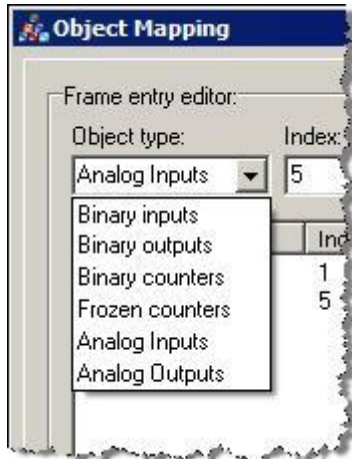
In the lower right hand corner of the Destination is an Object Map button.

This tool brings up the object editor:

The object editor is used to tell the Destination where to put data received from unsolicited slave commands and/or responses from a the master's data get commands.



In this case above, Any Analog Input (object) data from register 1 through 4 will be placed in the RemoteAIReal 1 through RemoteAIReal 4 respective of index. The same can be added for the following types:



More than one object of the same data type can be defined. The only rule is that indexes of the same data type cannot overlap. For ICL controllers the Register Data Type, Object Type and Index must match the value defined in the Network Message Link of the slave. Other manufacturers must supply class definitions and indexes so that you can configure these objects correctly.

If you are not using Class data on the slave side, then only static data is necessary to poll data out of a slave. The Network Message Links in the slave need simply be set to Class "None".

Creating DNP 3 Master Events

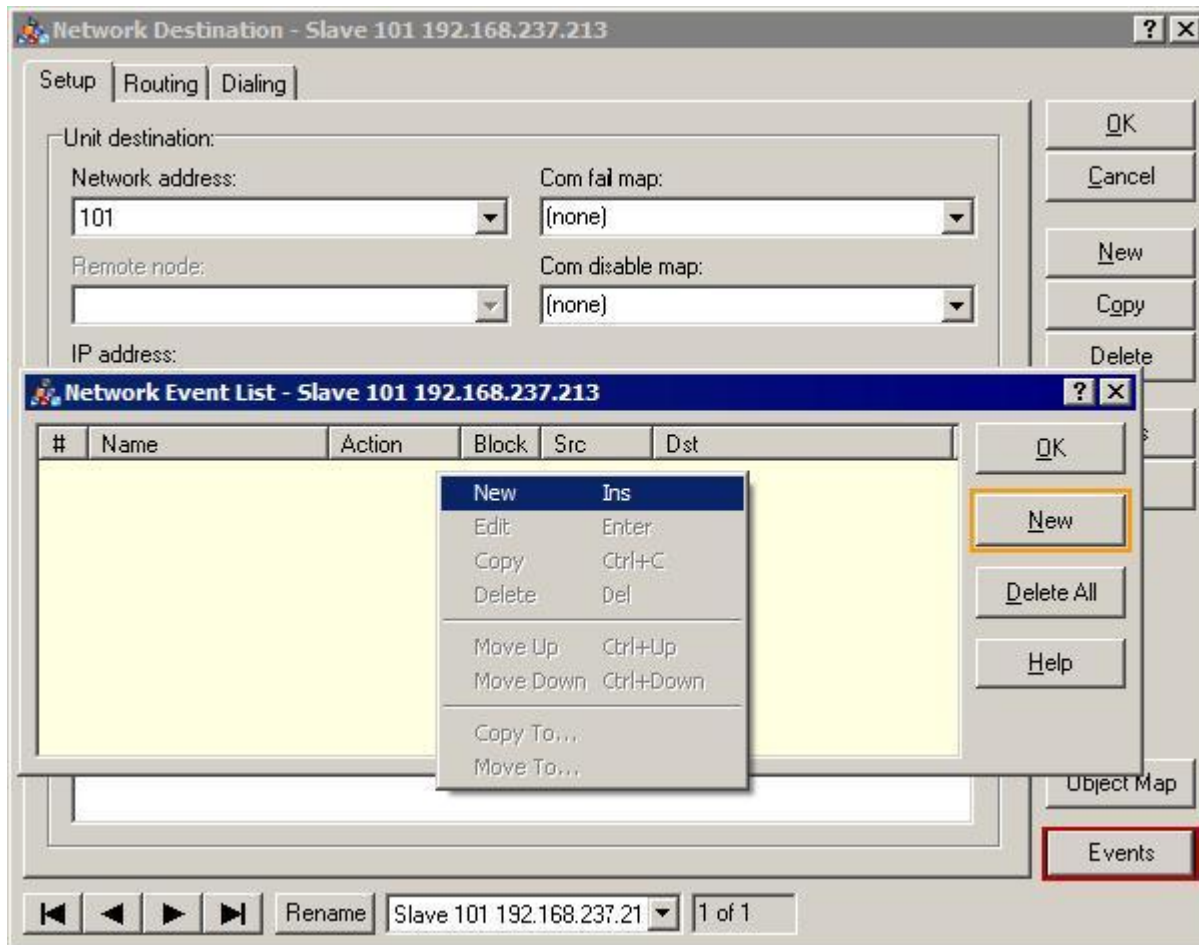


See *The ScadaBuilder Hierarchy* (on page 73).

Once a basic Network Session and Network Destination have been set up, you're ready to set up the Network Events for your Master.

Click on the "Events" button in the lower right-hand corner of the Network Destination window to bring up a "Network Event List" window.

To create an event, click on "New".



The resulting Network Event window has an optional field for the event name. Naming your Network Events makes it easier to understand what the events do and makes them easier to reference in the future for program maintenance.

Event | Activation | Options

Event name:
 Event name (optional):
 Read Real Inputs

Event message:

Action:
 Read

Source:
 Analog Inputs (static)

Index:
 (remote side)

Destination:

Index:

Block size (or select last register of block):
 [all] (all objects)

Click the 'Display' button to refresh list.

DNP 3 master events are different than every other protocol that ICL controllers support. There are many special features about DNP 3 that ICL controllers will support rather than just reading and writing data. For our first example, we will just read the static Analog Input data with the objects configured from the previous section. When the event is triggered and the message is received the data is placed according to the previous Master Object and Slave Network Message Link setup.

The event can be triggered manually, by time or by another Trigger. See **Network Event Activation** (on page 259) section for more details on triggering.

In this case the Master wants to read the analog inputs of the slave's static data. Any Network Message Link data configured on the slave as static analog input will return the data to be placed according to the master's object configuration.

Slave Network Message Link Configuration:

Message link:

Start register:
 UIReal1 (101)

Block size (or select last register of block):
 4

Group/variation:
 Analog Inputs (32-bit no flags)

Index:
 1

Event data:

Event class:
 (none)

Threshold (input deadband):
 1

☐ Report time with event data.

Local Registers: (Reals)	Msg Registers
101 (UIReal1)	Analog Inputs 1
102 (UIReal2)	Analog Inputs 2
103 (UIReal3)	Analog Inputs 3
104 (UIReal4)	Analog Inputs 4

Reg count: 4 Byte count: 16

Master Object Mapping Configuration:

Object Mapping dialog box configuration:

Frame entry editor:

Object type: Analog Inputs Index: 1 Start register: RemoteAIReal1 (201) Block: 4 Add

Object type	Index	Start Register	Block
Analog Inputs	1	RemoteAIReal1 (201)	4

Buttons: OK, Cancel, Help

With the above configuration, the data from Reals 101 - 104 of the slave will be polled for and copied to Reals 201 - 204. of the master.

DNP3 Slave Session configuration dialog box:

Message link configuration (top):

Start register: UIReal1 (101)

Block size (or select last register of block): 4

Group/variation: Analog Inputs (float w/ flags) Index: 1

Event data:

Event class: (none) Threshold (input deadband): 1

Report time with event data: ☐

Message link configuration (bottom):

Start register: UIInt32_1 (301)

Block size (or select last register of block): 4

Group/variation: Analog Inputs (32-bit w/ flags) Index: 5

Event data:

Event class: (none) Threshold (input deadband): 1

Report time with event data: ☐

Local Registers: (Integers)

301 (UIInt32_1)	Analog Inputs 5
302 (UIInt32_2)	Analog Inputs 6
303 (UIInt32_3)	Analog Inputs 7
304 (UIInt32_4)	Analog Inputs 8

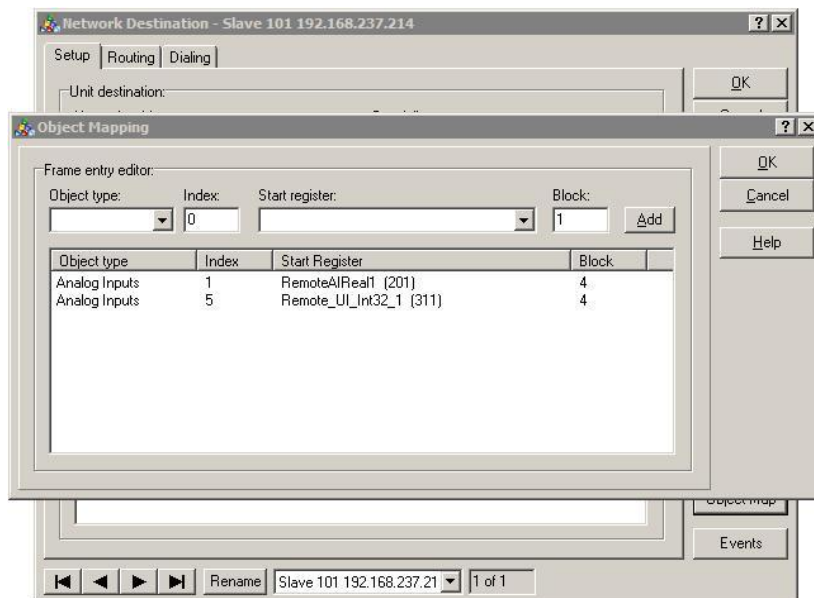
Msg Registers

Analog Inputs 1
Analog Inputs 2
Analog Inputs 3
Analog Inputs 4

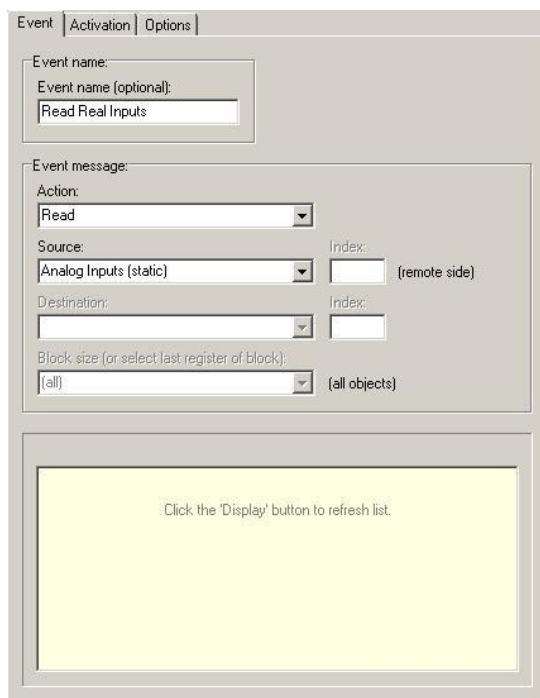
So let's add to this example so that the DNP Slave will return more than one data type. In the SlaveIP node we add a second Network Message Link That is an Analog Input 32-bit w/flags starting at index 5.

Slave Node:

On the DNP 3 master side in the object dialog, we add an object to handle where to put the data. We can have as many objects per destination as we want, but the data type and the index cannot overlap so we place the data this way:



The same Network Event that we used before will get both sets of data:



The Object list above tells the master where to put the received data, the slave when polled simple response with all analog input data configured in the Network Message Links. The data is sent to the master in two objects of different data types and indexes and the master places the data where it is told to from the Object List. The Object type of organization seems a lot of work for a simple data transfer so let's do another example and configure the master and slave to do a boolean event as well:

Adding a second data type

This Network Message Link will allow any master to read Binary Input (status) 1 through for and get the data from DI1 - DI4. Slave Message Links for Static Binary (status).

Message link:

Start register:

Block size (or select last register of block):

Group/variation: Index:

Event data:

Event class: Threshold (debounce x100 ms):

☐ Report time with event data.

Local Registers: (Booleans)	Msg Registers
20601 (DI1)	Binary Inputs 1
20602 (DI2)	Binary Inputs 2
20603 (DI3)	Binary Inputs 3
20604 (DI4)	Binary Inputs 4

We need to add some boolean registers to place the data and to add to the master the reciprocal object in the master's Object list for Binary Inputs:

Network Destination - Slave 101 192.168.237.214

Setup | Routing | Dialing

☐ Unit destination:

Object Mapping

Frame entry editor:

Object type: Index: Start register: Block:

Object type	Index	Start Register	Block
Analog Inputs	1	RemoteAIReal1 (201)	4
Analog Inputs	5	Remote_UI_In32_ (311)	4
Binary inputs	1	Remote_Bool_1 (401)	4

This requires the addition of a second Network Event in the master's slave Destination:

Event | Activation | Options

Event name:
Event name (optional):
Read Remote Binary Inputs

Event message:

Action:
Read

Source:
Binary inputs (static)

Index:
(remote side)

Destination:
(all objects)

Block size (or select last register of block):
(all)

With these configurations we can now read three data types with just two events.

Creating DNP 3 Master Class Events

A Class read event can make better use of the Network Event / Master Object Interface. The Slave configuration is the same if you are just reading static data. For the three Network Message Links on the slave that we created earlier, we are going to use a different kind of Network Event called a Class event. This Event will get all variations of a particular class (in this case Class 0). If a slave's Network Message Link is set to Class:"None", then it is automatically classified as Class 0 static data.

Using only one event on the master, we are going to get all Class 0 data:

Event | Activation | Options

Event name:
Event name (optional):
Read All

Event message:

Action:
Read

Source:
Class 0 (static)

Index:
(remote)

Destination:
(all objects)

Block size (or select last register of block):
(all)

From here we get all three data types from the slave:

The image shows three overlapping screenshots of a DNP3 configuration tool, likely for a slave device. The top-left screenshot shows the 'Analog Inputs (float w/ flags)' configuration. The top-right screenshot shows the 'Analog Inputs (32-bit w/ flags)' configuration. The bottom-left screenshot shows the 'Binary Inputs (status)' configuration. A tree view on the left shows the configuration structure.

Tree View:

- Ethernet
 - DNP3SlaveSession
 - UI Input Reals
 - UI Input Int32
 - Binary Status DI's

Configuration Screenshots:

Top-Left: Analog Inputs (float w/ flags)

- Group/variation: Analog Inputs (float w/ flags)
- Index: 1
- Event data:
 - Event class: (none)
 - Threshold (input deadband): 1
 - ☐ Report time with event data
- Local Registers: (Reals)

101 (UIReal1)
102 (UIReal2)
103 (UIReal3)
104 (UIReal4)
- Msg Registers

Analog Inputs 1
Analog Inputs 2
Analog Inputs 3
Analog Inputs 4

Top-Right: Analog Inputs (32-bit w/ flags)

- Group/variation: Analog Inputs (32-bit w/ flags)
- Index: 5
- Event data:
 - Event class: (none)
 - Threshold (input deadband): 1
 - ☐ Report time with event data
- Local Registers: (Integers)

301 (UIInt32_1)
302 (UIInt32_2)
303 (UIInt32_3)
304 (UIInt32_4)
- Msg Registers

Analog Inputs 5
Analog Inputs 6
Analog Inputs 7
Analog Inputs 8

Bottom-Left: Binary Inputs (status)

- Group/variation: Binary Inputs (status)
- Index: 1
- Event data:
 - Event class: (none)
 - Threshold (debounce x100 ms): 0
 - ☐ Report time with event data
- Local Registers: (Booleans)

20601 (DI1)
20602 (DI2)
20603 (DI3)
20604 (DI4)
- Msg Registers

Binary Inputs 1
Binary Inputs 2
Binary Inputs 3
Binary Inputs 4

In one transaction, typically one poll message and one response, the master is able to get all the static (a.k.a) live data configured in the slave.

Supported DNP 3 Master Commands

This is a list of supported command from the DNP 3 Master Network Event.

Action	Supported Data (Variants)
Read Read static and event data from slave.	Classes Class 0 (static) Class 1 - 3 (event) Static Objects Binary Inputs (static) Binary Outputs (static) Binary Counters (static) Frozen Counters (static) Analog Inputs (static) Analog Outputs (static) Event Objects Binary Inputs (event) Binary Outputs (event) Analog Inputs (event)
Select Select a point for operation (make it ready for the Operate command).	Binary Outputs Analog Outputs (32 bit) Analog Outputs (16 bit) Analog Outputs (Float)
Operate Actuate Selected operation.	Binary Outputs Analog Outputs (32 bit) Analog Outputs (16 bit) Analog Outputs (Float)
Direct Operate Operate without Select command with acknowledgement.	Binary Outputs Analog Outputs (32 bit) Analog Outputs (16 bit) Analog Outputs (Float)
Direct Operate (NA) Operate without Select command with no acknowledgement.	Binary Outputs Analog Outputs (32 bit) Analog Outputs (16 bit) Analog Outputs (Float)
Immediate Freeze Freeze Counter with acknowledgement.	Binary Counters (static)
Immediate Freeze (NA) Freeze Counter with no acknowledgement.	Binary Counters (static)

Freeze and Clear Counter with acknowledgement.	Binary Counters (static)
Freeze and Clear Counter with no acknowledgement.	Binary Counters (static)
Time Synchronization	Real Time Clock
Cold Restart	Reboot Controller
Warm Restart	Restart Application

Slave Configuration To Retrieve Event Data

ICL DNP 3 Slaves have the ability to return unsolicited data, Event data and Event Data with time stamps. This is useful for SCADA systems that support back-filling trends and data logs.

When communications goes down and is restored or the network is slow in getting around to all slaves, the ICL DNP 3 slave can queue multiple event data points in an internal RAM buffer. When polled for Class data or specific object types with time stamps, the slave can return the data back to the master SCADA system to fill in any missing time.

These options are configured in the DNP 3 Slave Protocol Network Session DNP tab. In order to use these features, at least one Class (x) interface must be enabled.

The screenshot shows the 'DNP' tab in a configuration window. It is divided into two main sections: 'Unsolicited responses' and 'Outstation settings'.

Unsolicited responses:

- ☒ Enable Class 1:
 - Hold time: 60 (sec)
 - Hold count: 10
- ☐ Enable Class 2:
 - Hold time: 60 (sec)
 - Hold count: 10
- ☐ Enable Class 3:
 - Hold time: 60 (sec)
 - Hold count: 10
- ☐ Send initial unsolicited response at startup ...
- ☒ Include static data.

Outstation settings:

- Event buffer size: 1024
- Master address: 100
- Master IP address: msgSlaveIPAddress (1)
- Operate timeout: 10 (sec)
- Time sync interval: 60 (min)
- Response confirmation: Event data & multi-fragment responses.

If you want to use DNP 3 Event data then the Event Buffer Size should be considered. This number represents the number of events that will be stored in each of the Classes selected above. You can select up to 3 Classes. The resulting change events are stored in RAM. Should communications go down between the SCADA system and this unit, the data will be stored until communications is restored.

If you want the data to come back with absolute (date and time of day) time stamps, then the Time Sync Interval should be set. This period represents how often the slave will ask the master for a time sync message to be sure that all units in the system are synchronized to the same master clock.

Configuring the Network Message Link For Event Data

The Network Message Link has the ability to use event data registers with time stamp.

A Class number must be selected.

The registers must be configure "With Flags" or w/flags.

The Report Time With Event Data check box checked.

The Threshold (Input Deadband) field set.

Local Registers: (Reals)	Msg Registers
101 (UIReal1)	Analog Inputs 1
102 (UIReal2)	Analog Inputs 2
103 (UIReal3)	Analog Inputs 3
104 (UIReal4)	Analog Inputs 4

Once the Network Message Links configuration is setup for the registers of interest, the slave will record event data with every change based on the Threshold configuration. When a SCADA master system asks for data with time or the Class data specified, the slave will respond with all event data it has buffered since the last poll and then will clear the data if specified from the master.

Getting Event Data With a SCADA System

SCADA systems vary widely in how they handle trending and data logging. Back-filling data is possible for any point that supports the functionality over a DNP 3 link.

Trihedral VT Scada Example

For a Boolean Input point you must configure the tag as a Digital Status (not a Digital Input).

For a Boolean Output point you must configure the tag as a Digital Control (not a Digital Output).

For a Analog Input point you must configure the tag as a Analog Status (not an Analog Input).

For a Analog Output point you must configure the tag as a Analog Control (not an Analog Output).

Asking For Event Data

To ask for the static value for an Analog Status(using our example above) you must configure a trend or data log tag as:

32/3/1:NS

32 represents the Object Type: Analog Input Event (meaning with time stamp)

3 represents the Any Variant With Time.

1 represent the Analog Input register index.

NS (VT Scada Configuration) use polling engine for polling interval.

Asking For Static Data (Current Value)

To ask for the static data or live value, a VT Scada example would look like this:

30/1/1:NS

30 represents the Object Type: Analog Input Static (meaning with time stamp)

1 represents the Any Variant Without Time.

1 represent the Analog Input register index.

NS (VT Scada Configuration) use polling engine for polling interval.

For details on supported types and variants, see the ***DNP 3 Compliance Document***
http://www.iclinks.com/public_ftp/TechSupport/DNPPProfile/PinnacleDnpProfile.htm.

Ethernet IP Protocol

Ethernet IP is used to talk to Allen-Bradley, SLC5, Micro Logix and Control/Compact Logix using Ethernet (and TCP/IP) media layer.

Currently this is only a partial implementation as communicating by tag name configurations is not supported.

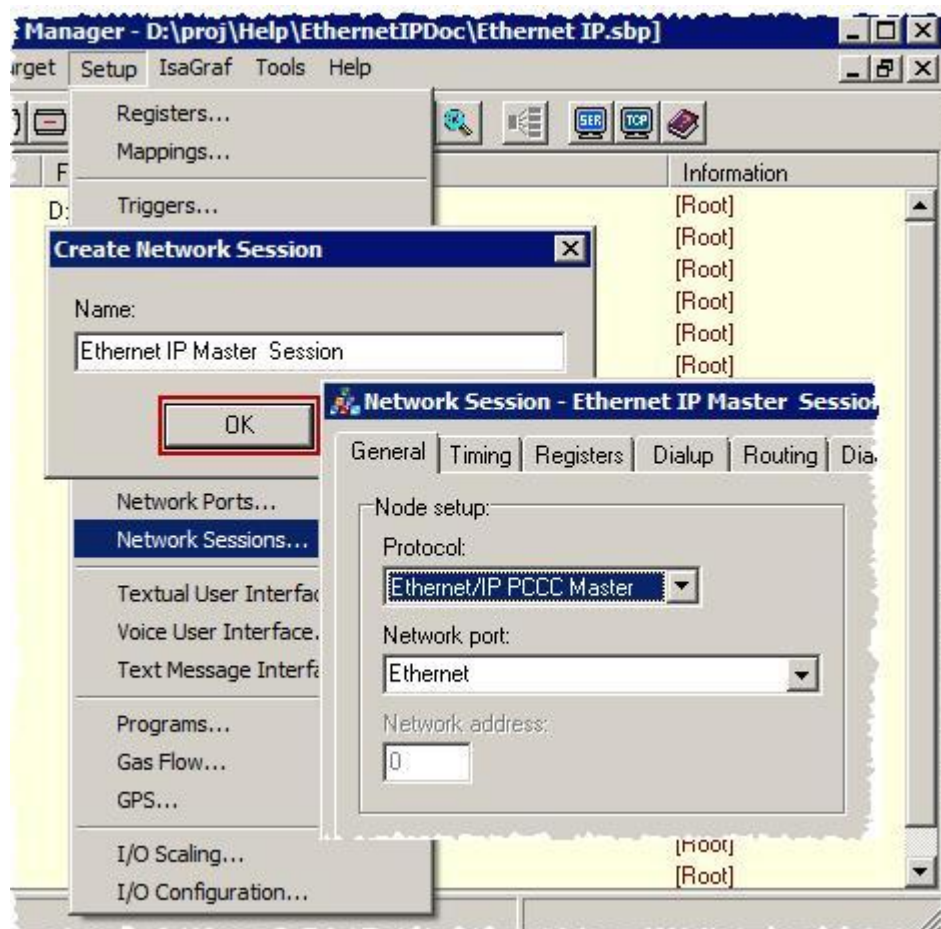
ICL Pinnacle and later controllers support both Ethernet IP Master and Ethernet IP Slave Protocol Interfaces over Ethernet.

Creating An Ethernet IP Master Interface



See *The ScadaBuilder Hierarchy* (on page 73).

The Ethernet Port on a Pinnacle and later controller should already be setup as required for downloading and debugging programs.



To create an Ethernet IP Master session, select the **Setup | Network Sessions...** menu. A dialogue window will pop up for naming the session. Accept the default name or enter a new one. If you already have at least one Network Session, you can also simply click on the “New” button on the right-hand side of the Network Session window.

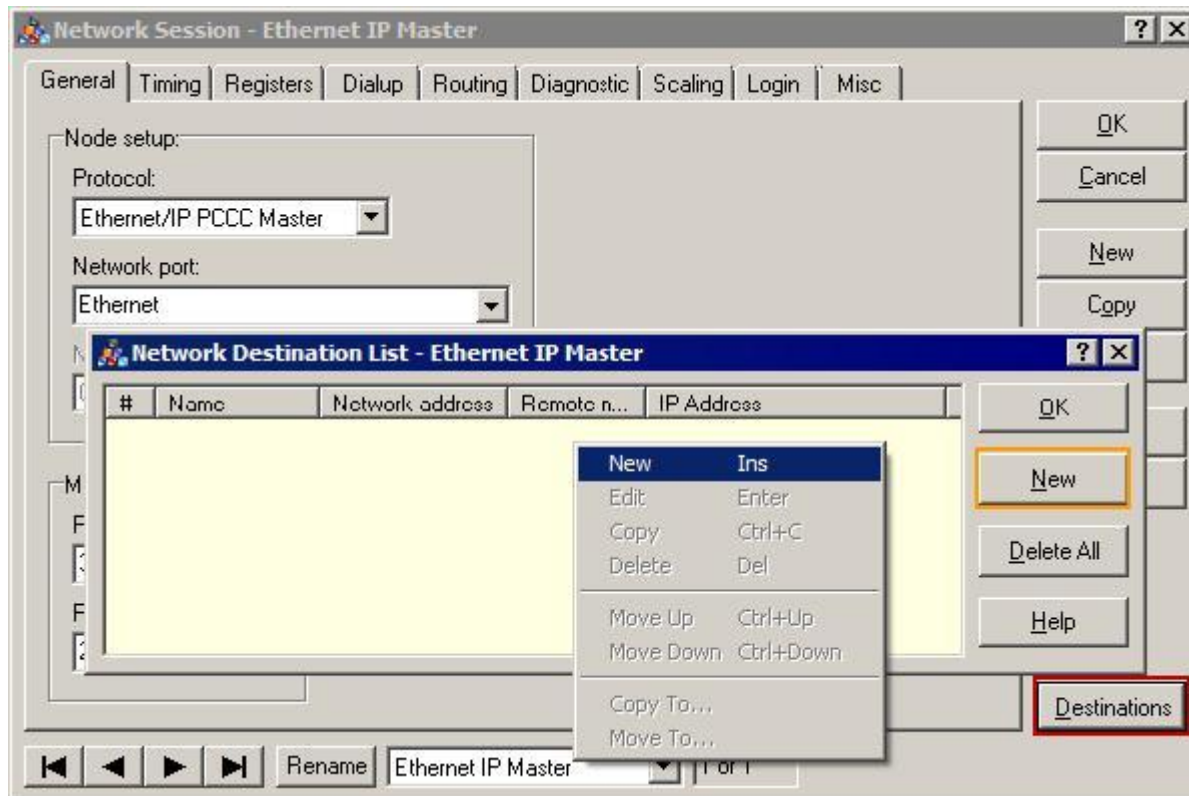
Network Port - After selecting the protocol, choose the Network Port which will be "Ethernet". This will be the only compatible port to the Ethernet IP protocol.

Creating An Ethernet IP Master Destination

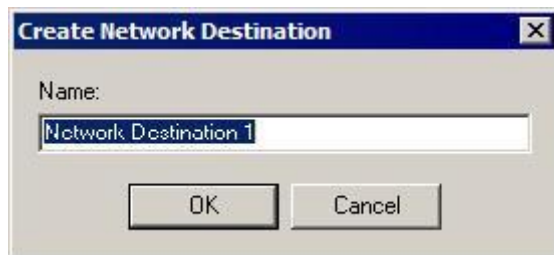


See *The ScadaBuilder Hierarchy* (on page 73).

Each remote slave that the Ethernet IP Master Network Session needs to talk to must be setup as a Network Destination. To create a Network Destination, Open the Ethernet IP Session and click on the Destinations button in the lower right hand corner to bring up the Network Destination List.



Right click in the list area or click on the New button to start a new Network Destination. You should see this dialog:



Use the default name or give it a new one. You can name it anything you like. If you do use a name it is recommended that the Site name be used and the slave address and IP address be included so the Destination is readily identified.

Click Ok .

At minimum, enter the Network Address and the IP address of the remote Ethernet IP slave.

You will also have to choose the Messaging Mode. For PLC5 compatible devices, choose the PL5E Messaging option. For SLC500 series and Compact/Control Logix, choose the SLC505 Messaging option.

Network Destination - EIP Slave 216

Setup | Routing | Dialing

Unit destination:

Network address: 1 Com fail map: (none)

Messaging mode: SLC505 Messaging Com disable map: (none)

IP address: 192.168.237.216

Com statistics:

Select Register: Map

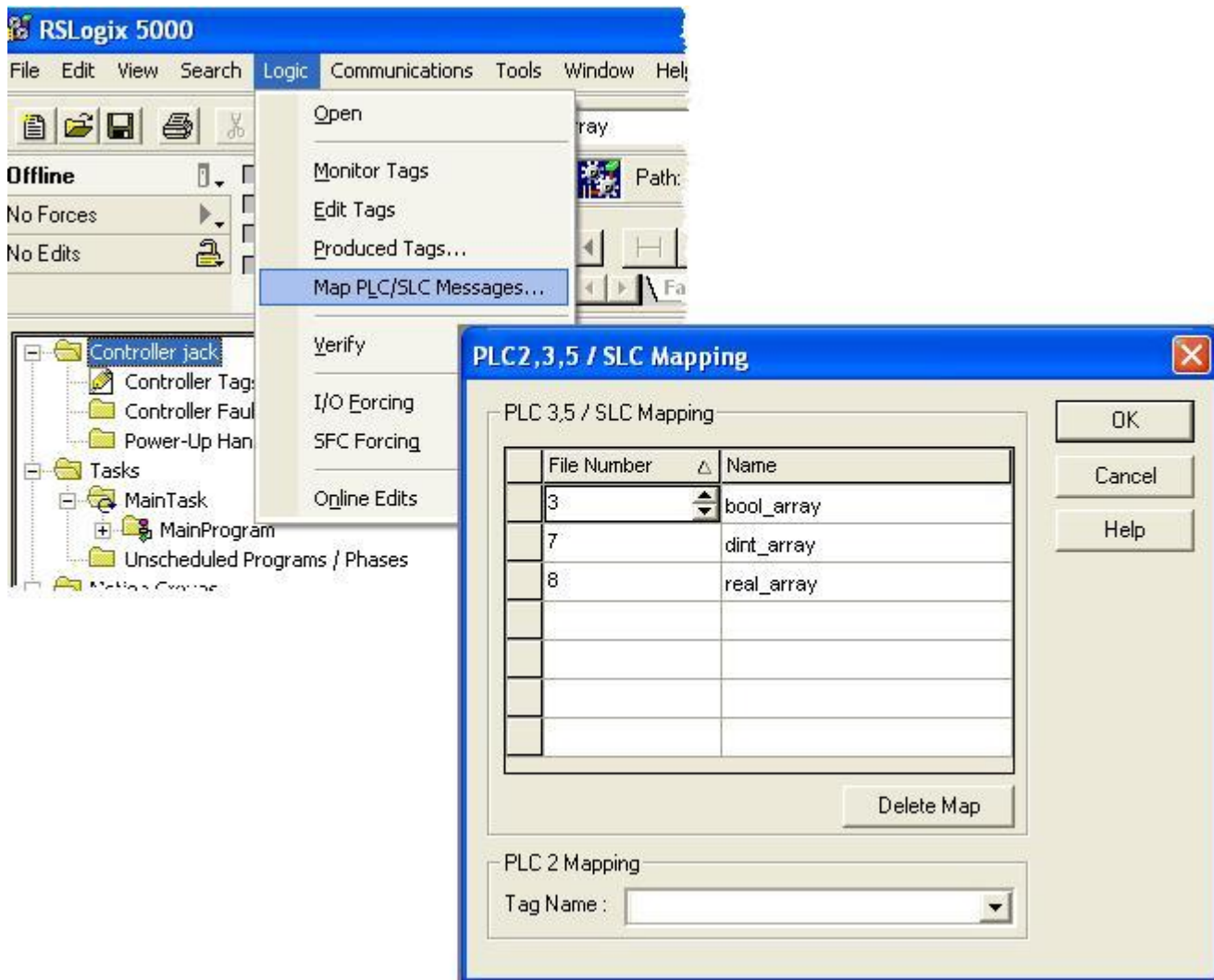
Statistic	Register

OK Cancel New Copy Delete Notes Help Events

Navigation buttons: [Previous] [Next] [First] [Last] [Rename] EIP Slave 216 1 of 1

Optionally, you can also map the Com Fail flag to a boolean registers to monitor the general health of the slave. You can also map the Com Disable boolean and communications statistics for more control and monitoring capability. See **Address Com Statistics** (see "Com Statistics" on page 247) for more information.

For Control/Compact Logix Etherent IP slaves, you will need to create data type arrays on the remote units (if not already used for your remote program). In the RS Logix program, you will need to setup the Ethernet IP protocol in SLC mode, and use the Logic / MapPLC/SLC Messages... option to assign interface numbers (File Numbers) to individual data arrays depending on your interfacing needs.



If you need to write directly to I/O, the PLC program should be modified to move data from the configured data arrays to the I/O desired. The Ethernet IP Master Session does not support writing to I/O directly to a Compact/Control Logix controller.

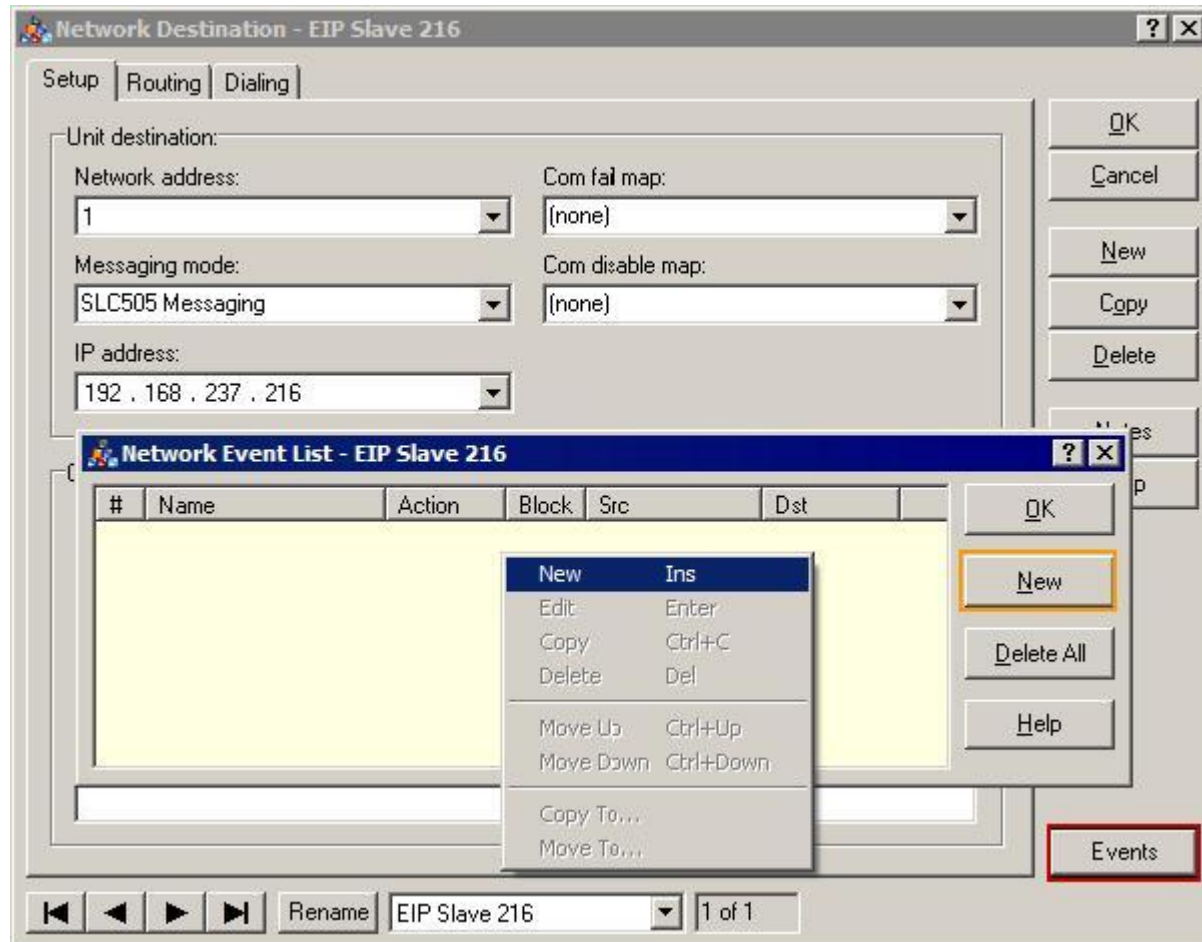
Creating Ethernet IP Master Network Events

Once the basic Network Session and Network Destination have been set up, you're ready to set up the Network Events for your Master.

Click on the "Events" button in the lower right-hand corner of the window to bring up a "Network Event List" window.

To create an event, click on "New" or right click in the list and select New.

The resulting Network Event window has an optional field for the event name. Naming your Network Events makes it easier to understand what the events do in the future.



Event | Activation | Options

Event name:
Event name (optional):
Read EIP AI's

Event message:
Action:
Read

Source: N7 (integer) Element: 1 (remote side)

Destination: RemoteAI1 (601) Element: (local side)

Block size (or select last register of block): 20 (destination register block)

(DF1 References)	(Booleans)
File: N7, Elem: 1	601 (RemoteAI1)
File: N7, Elem: 2	602 (RemoteAI2)
File: N7, Elem: 3	603 (RemoteAI3)
File: N7, Elem: 4	604 (RemoteAI4)
File: N7, Elem: 5	605 (RemoteAI5)
File: N7, Elem: 6	606 (RemoteAI6)
File: N7, Elem: 7	607 (RemoteAI7)
File: N7, Elem: 8	608 (RemoteAI8)
File: N7, Elem: 9	609 (RemoteAI9)
File: N7, Elem: 10	610 (RemoteAI10)

First we need to define the type of data transfer. An event can read registers, write registers, or probe a slave. Select an action (read, write or probe) in the appropriate fields.

For efficiency, all data transfers are done in blocks of multiple registers. Enter the number of registers to be transferred as the "Block size".

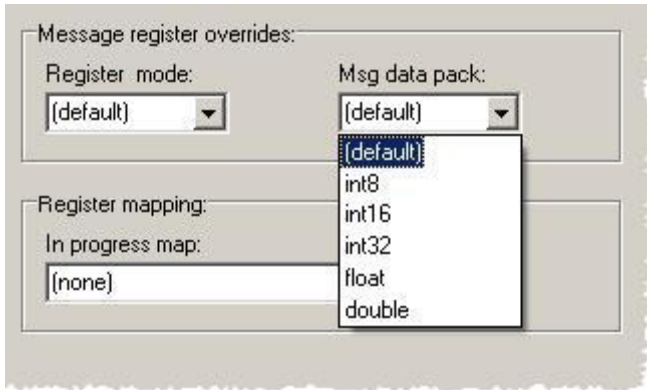
For a "read" event, data will be transferred from the slave's B3, N7 or F8 registers (or you can enter your own file number and select the data type on the Options tab) to matching registers in the controller. Select the Source data type and Index (register number) in the remote slave, and the FIRST register of the block that the data is to be transferred to in the controller.

For a "write" event, the Source will be the first register of a block of registers in the controller and the destination will be the corresponding first register in the remote slave that the data will be transferred to.

Click on the Activation tab. See *Network Event Activation* (on page 259) section for more details.

Event Options

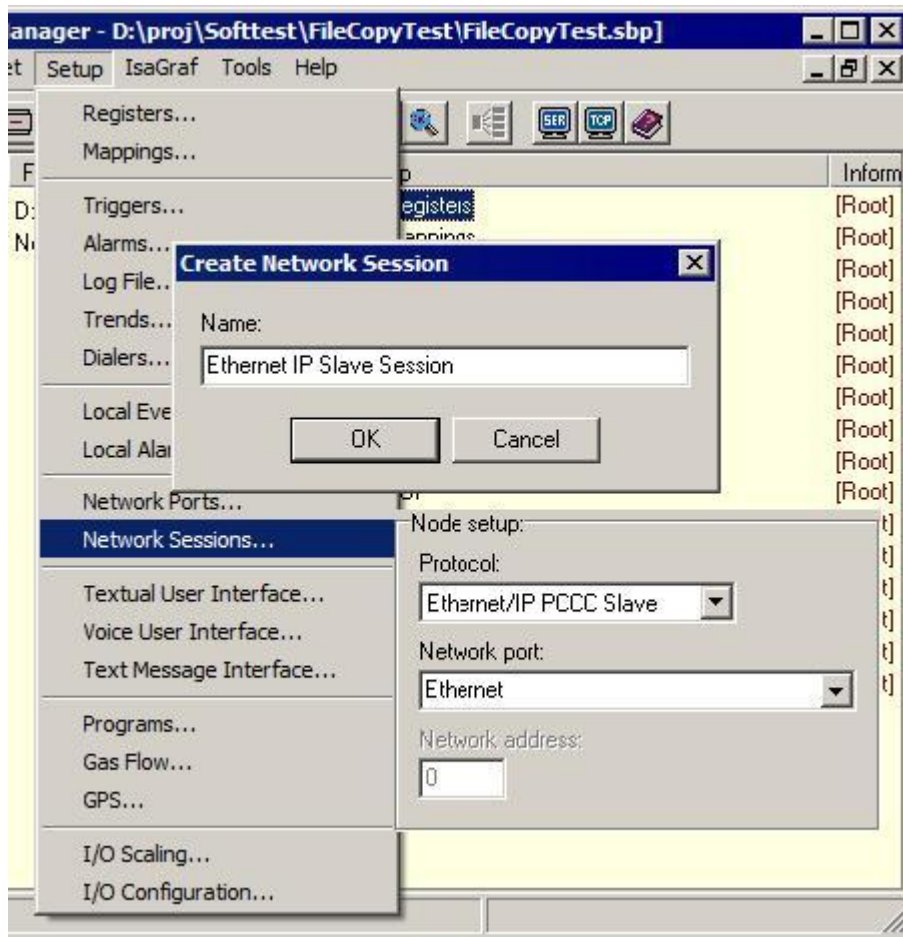
Under Ethernet IP, the Network Event Options tab is an important one. Since any bank on the AB side can be a different data type, we must choose how to interpret the returned (or sent) message data.



The (default) selection will simply use the local register type on the ICL controller. If the local event register are Reals (floating point), then the "float" option will be used. If the local event registers are Integers (32 bit) then Int 32 will be used.

If any other Msg Data Pack option is used, it will override this default. A common thing to do with N7 type message is to set this Msg Data Pack option to "int16" to convert the 16 bit value coming in from the AB's N7 bank to a 32 bit or floating point value (depending on the local register type).

Creating An Ethernet IP Slave Interface



A Ethernet IP Slave session is easier to configure than a Master because a Slave simply needs to listen and respond to requests and commands from the Master. There's no event configuration!

To create a Ethernet IP Slave session, select the **Setup | Network Sessions...** menu. A dialogue window will pop up for naming the session. Accept the default name or enter a new name. If you already have at least one Network Session already, you can simply click on the "New" button on the right-hand side of the Network Session window.

Protocol - Once you have a new Network Session window, select the protocol Ethernet/IP PCCC Slave.

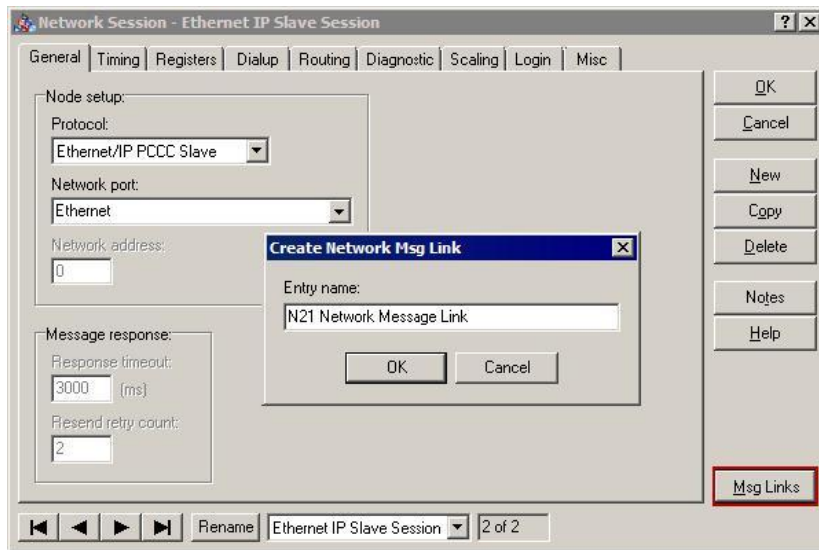
Network Port - Ethernet is the only supported Network Port for any Ethernet IP interface.

Creating Ethernet IP Network Message Links

For an Ethernet IP master to talk to and ICL Ethernet IP Slave session, it is necessary to setup Network Message Links to tell the Ethernet IP Slave Session how to interpret incoming messages. Each file number and register range combination read from or written to from an Ethernet IP master must have a Network Message Link configured for it. There is no default data type/register handling built into a Ethernet IP Slave Session alone.

When configuring the AB or other master (including an ICL Ethernet IP Master), the SLC5 mode must be used to communication to the ICL Slave Session / Network Message Link interface.

In the lower right hand corner of the Ethernet IP Slave session is the `Msg_Links` button. Click on this button and give the new Network Message Links a name and Click OK.



Shown here is a sample Network Message Link configuration.

Enter the Start Register which selects what local registers the Network Message Link will operate on.

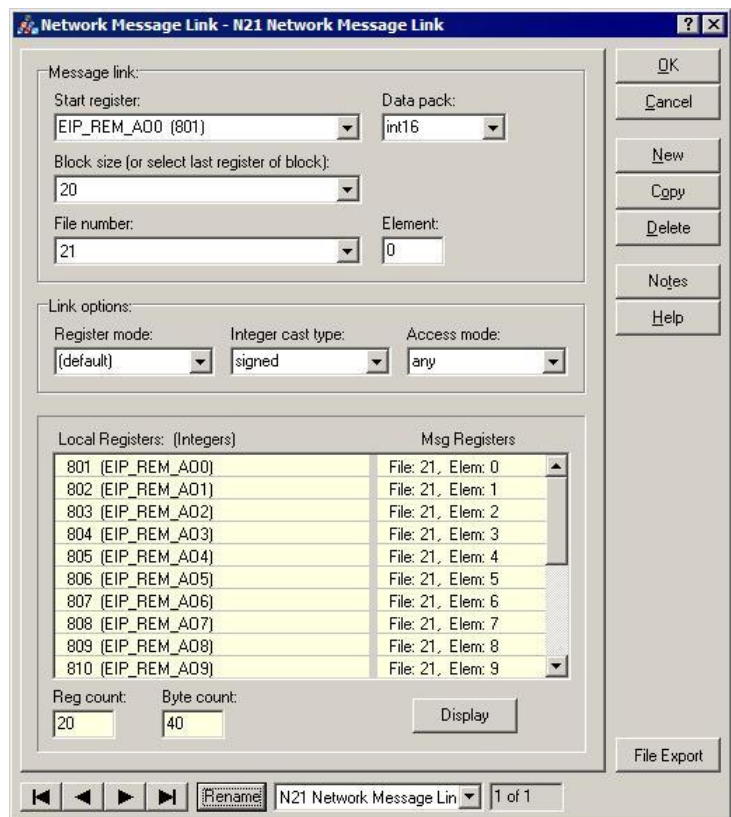
Select the Blocksize.

For the File Number, you have the option of selecting from default file numbers that will be received for the protocol but you can also manually select the file number desired.

Enter the Element number. This will be the number you enter along with the file number for data reads and writes from your Ethernet IP master.

The Data Pack mode tells the Slave Session how to translate the ICL local data registers (32 bit Integers or Real floating point) and transmit it back to or receive from the master. Essentially this is the data type the master will be writing to (read message) or reading from (write message) in its local file number banks.

Integer Cast type when set to "signed" tells whether to translate the sign bit to and from the local registers.



Hart Master Protocol



See *The ScadaBuilder Hierarchy* (on page 73).

The Hart Protocol, like Modbus, was originally a flowmeter communications medium for getting data over a two wire interface (the same wires carry the 4-20 mA signal). Today, Hart is used for standard networks but still retains the 4 - 20mA capability of old.



In order to use the Hart Master Protocol, you must have a serial to Hart format Modem.

ICL sells a Hart Modem that mounts internally in the Etherlogic and ScadaFlex Plus lines of controllers.

For Pinnacle and later models, a USB Hart Modem may be used.

For more information about to commands supported, see the **Command** (see "Command (Hart Supported)" on page 345) section later in this chapter.

Creating a Hart Master Interface

Some thought should go into how you want to set up your Hart Master. In its simplest form, the Master can continuously request data from Hart Slaves, and continuously write data to them. This is the easiest form of Master to create, but it is also the least efficient.



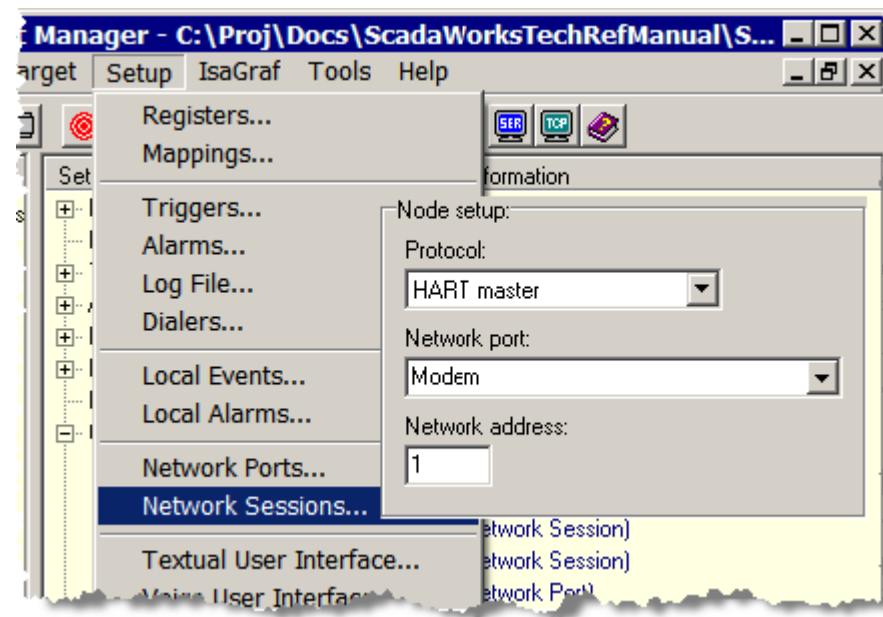
With a little more effort, it's possible to prioritize the frequency of the events that cause messages to be sent and use Triggers to send data intelligently when it changes. See Using Triggers for more details.

For example, remote outputs can be updated when outputs need to be changed instead of constantly writing the same output data over and over. This is accomplished by defining Triggers. A single Trigger can look for a change in a block of registers. When a change occurs, the Trigger can cause a Hart message to be sent that updates the remote outputs. Triggers can be added at any time as needed.

You will need to set up a Network Port that defines the basic hardware level communications elements such as baud rate, parity and hardware interface or you can simply use the defaults for now and change the Network Port to your liking later on. You will also need a Hart Modem on that serial port interface.



See *The ScadaBuilder Hierarchy* (on page 73).



To create a Hart Master session, select the **Setup | Network Sessions...** menu. A dialogue window will pop up for naming the session. Accept the default name or enter a new one. If you already have at least one Network Session, you can also simply click on the “New” button on the right-hand side of the Network Session window.

Select the protocol as Hart Master.

Select your Network Port

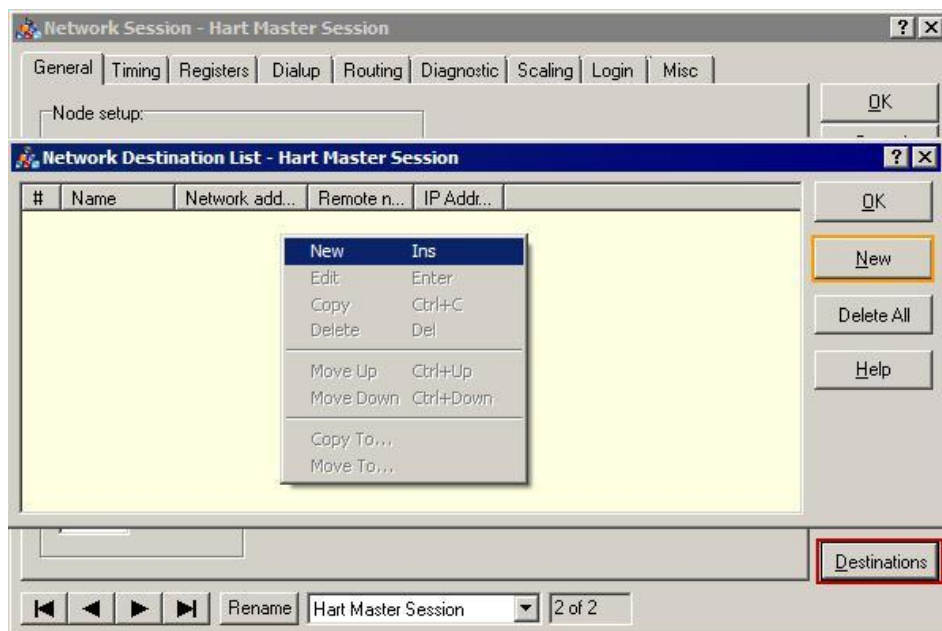
A Hart Master must be assigned an address between 1 and 255. You also have the option of using DIP switches on the Controller (EtherLogic or ScadaFlex Plus family) or a register for changing the slave address. This is accomplished through fields under the “Misc” Tab. If “Register” is chosen, you must select the tag name of an integer register and configure it as “retained” (nonvolatile).

For more details, see *Network Sessions Reference* (on page 218) section.

Creating a Hart Master Destination

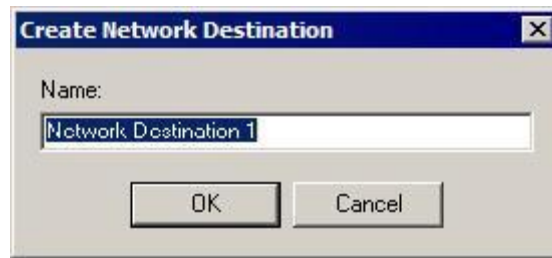


See *The ScadaBuilder Hierarchy* (on page 73).



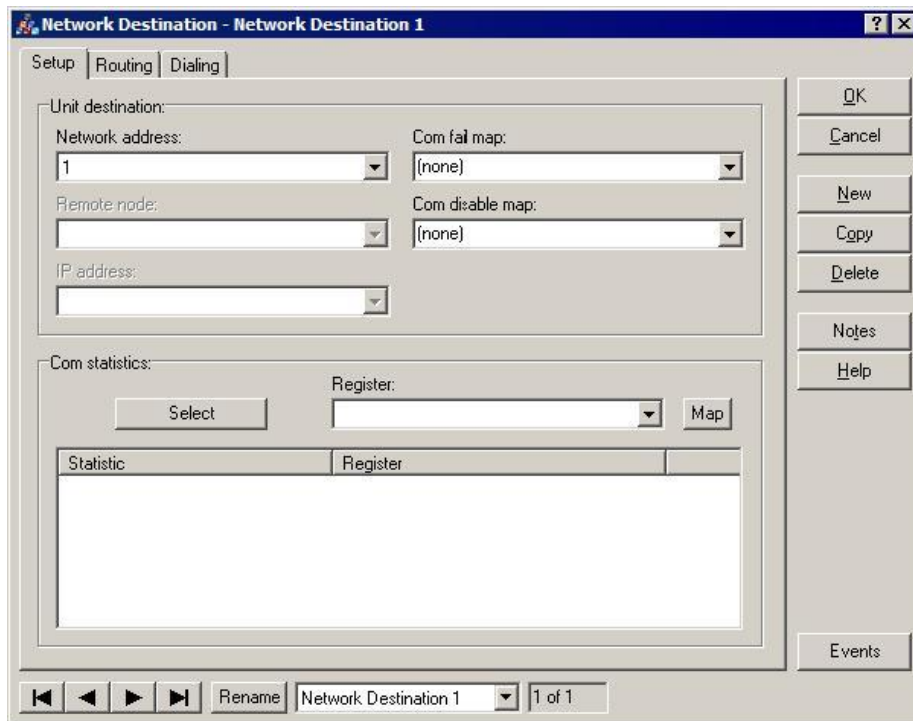
Each remote slave that the Hart Master Network Session needs to talk to must be setup as a Network Destination. To create a Network Destination, Open the Hart Master Session and click on the Destinations button in the lower right hand corner to bring up the Network Destination List.

Right click in the list area or click on the New button to start a new Network Destination. You should see this dialog:



Use the default name or give it a new one. You can name it anything you like. If you do use a name it is recommended that the Site name be used and the Hart Slave address be included so the Destination is readily identified.

Click Ok .



Optionally, you can also map the Com Fail flag to a boolean registers to monitor the general health of the slave. You can also map the Com Disable boolean and communications statistics for more control and monitoring capability. See *Address Com Statistics* (see "Com Statistics" on page 247) for more information.

Note: The Network Address field must still match the remote's Hart slave address for the data transfer to work.

Creating Hart Master Events

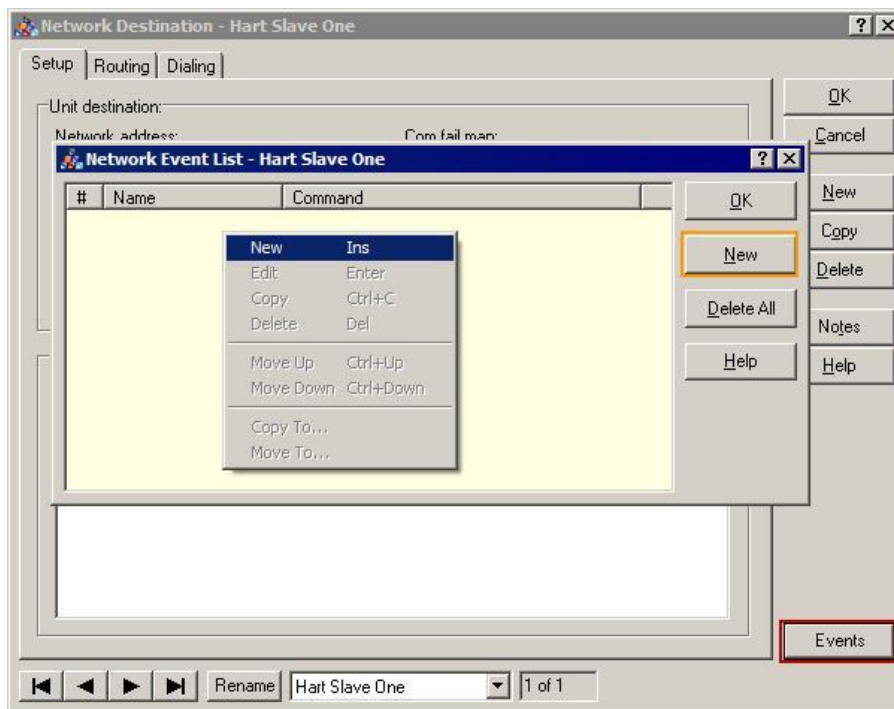


See *The ScadaBuilder Hierarchy* (on page 73).

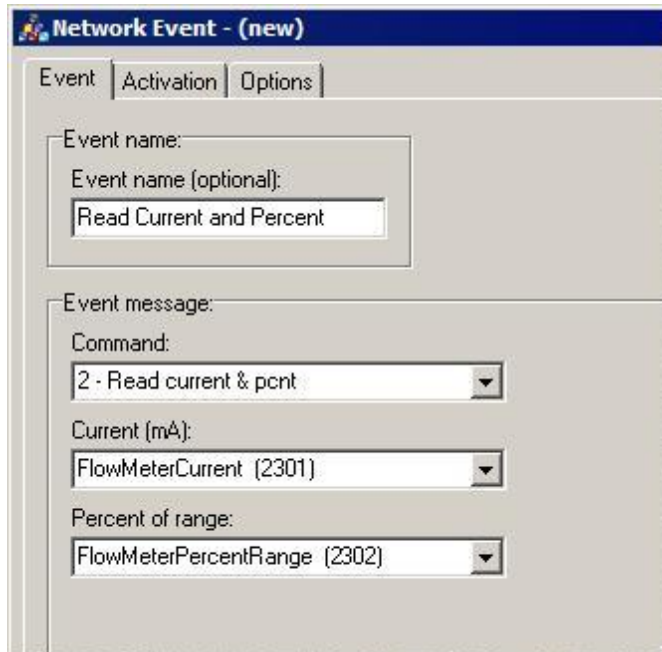
Once a basic Network Session has been set up, you're ready to set up the events for your Master.

Click on the "Events" button in the lower right-hand corner of the window to bring up a "Network Event List" window.

To create an event, click on “New”.



The resulting Network Event window has an optional field for the event name. Naming your Network Events makes it easier to understand what the events do in the future.



For a “read” event, data will be transferred from the slave’s registers to matching registers in the controller. Select the Source data type and Index (register number) in the remote slave, and the registers that are to be transferred to in the controller.

The supported commands are:

- 0 - Read unique identifier
- 1 - Read primary variable
- 2 - Read current & pcont
- 3 - Read current & dyn vars
- 6 - Write polling address (BC)
- 11 - Read UID from tag
- 12 - Read Message
- 13 - Read tag, desc & date
- 14 - Read PV sensor info
- 15 - Read device info
- 16 - Read assembly number
- 17 - Write message
- 18 - Write tag, desc & date (BC)
- 19 - Write assembly number
- 132 - Write setup var (SM-1020-0144)
- 130 - Write change var (Rotork)

For a “write” event, the Source will be the first register of a block of registers in the controller and the destination will be the corresponding first register in the remote slave that the data will be transferred to or from.



The Event message area will change as different commands are selected.

Click on the Activation tab.

See **Network Event Activation** (on page 259) section for more details.

Command (Hart Supported)

Hart Master Command tells the system what style of communication we are doing to the HART device.

The possible commands their assignable parameters and data types are:

Hart Supported Command Number	Parameter(s)
0 - Read Unique Identifier	Mfg ID Code (integer) Mfg Device Type (integer) Device ID Number (integer)
1 - Read Primary Variable	Units Code (integer) Primary Variable (integer) (real)
2 - Read Current and Percent	Current (mA) (integer) (real) Percentage of range (integer) (real)
3 - Read Current and Dynamic Variables	Current (mA) (integer) (real) PV Units code (integer) Primary variable (integer) (real) SV units code (integer) Second variable (integer) (real) TV units code (integer) Third variable (integer) (real) FV units code (integer) Fourth Variable (integer) (real)
6 - Write Polling Address	Polling address
11 - Read Unit ID from Tag	Tag Name (message) Mfg ID code (integer) Mfg device type (integer) Device ID number (integer)
12 - Read Message	Message (message)
13 - Read Tag Descriptor and Date	Tag Name (message) Descriptor (message) Day (integer) Month (integer) Year (integer)
14 - Read Primary Variable Sensor Info	Serial number (integer) Units code (integer) Upper limit (integer/real) Lower limit (integer/real) Minimum span(integer/real)
15 - Read Device Info	Alarm select code (integer) Transfer funct code (integer) Range units code (integer) Upper range (integer/real)

	Lower range (integer/real)
	Damping value (integer/real)
	Write protect code (integer)
	Private label code (integer)
16 - Read Assembly Number	Assembly number (integer)
17 - Write Message	Message (message)
18 - Write Tag, Descriptor and Date	Tag name (message)
	Descriptor (integer)
	Day (integer)
	Month (integer)
	Year (integer)
19 - Write Assembly Number	Assembly number (integer)
130 - Write Change Var (Rotork)	Change Variable (integer/real)
132 - Special Command for SM-1020-044	Index (user entered)
Write Setup Variable.	Units Code (integer)
	Setup variable (integer/real)

Each command requires different source and destination variables to be configured.

Bricknet Protocol



See *The ScadaBuilder Hierarchy* (on page 73).

Bricknet is ICL's own peer-to-peer SCADA protocol, originally developed for the ICL-4300 "PC-in-a-Brick" controller. It has the ability to handle more data types than Modbus or DF1 including single bits, 8-bit, 16-bit and 32-bit integers, floating point values, and strings. It also has framing characters to make it more immune to noisy data links, and includes a full message routing capability (aka Store and Forward).



With Bricknet, each controller is aware of the other controllers' registers. There is no need to remember register indexes, nor is there any need for weird data conversions or register offsets. All data is transferred "as is".

Even with these enhancements, it has a very low message overhead to accommodate slow data links. Most importantly, Bricknet is a peer-to-peer protocol so that any node can initiate a message. This means that a very efficient network can be built using a "transmit on exception" architecture that can eliminate the need for polling which improves responsiveness in large system configurations.

Although Bricknet was originally developed for ICL controllers, it is openly published and may be freely used by anyone.

Bricknet can also be used over a telephone dialup link.

It also supports File Transferring which means log files can be retrieved and programs can be downloaded and restarted over a serial data link--*without any interruption in network service.*

While they cannot initiate commands, all ICL RTU's including Picobrick, Microbrick, MAXIO, ScadaFlex RTU, Sprites and other Sentry Series RTU's can respond to Bricknet commands.



Bricknet is a peer-to-peer protocol so that any node can initiate a message. Think of it as having a Master and Slave protocol on the same port!

Creating a Bricknet Interface

Because Bricknet is a peer-to-peer protocol, there are more choices on how a system is designed. It's possible to set up a Master-Slave type relationship just like Modbus; for example, a pump controller that continually polls for level information from tank sites. On the other hand, a more efficient system can be created having the tank sites send level information to the pump controller only when the tank levels change, and periodically to make sure that the Pump Controller knows that the tank sites are still alive. Because time is not wasted by constant polling, response times are faster and less network bandwidth is used; ideal for slower networks like radio based systems.

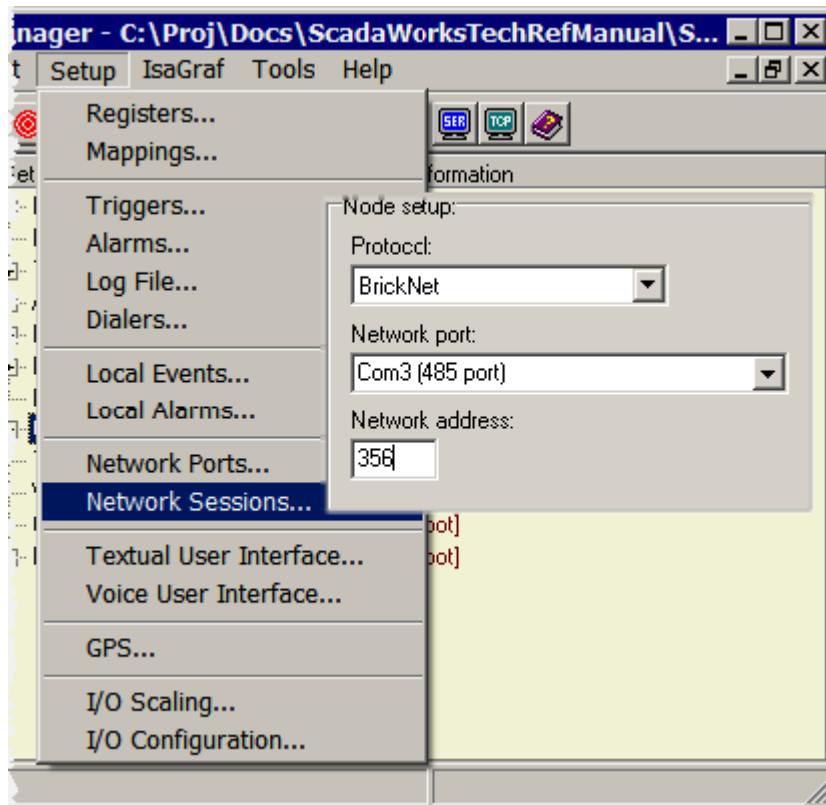
In the above example, the tank sites would use "Triggers". One of the available Trigger types senses a level change of a preset amount (Delta Trigger). When the tank level changes, the Trigger is set, causing a message to be sent to the pump station. Another Trigger might be used for an intrusion alarm, immediately sending a message when a discrete input signal from a motion sensor detects an intruder (State On Trigger). *See Using Triggers for more details.*

You will need to set up a Network Port that defines the basic hardware level communications elements such as baud rate, parity and hardware interface (RS-232, RS-485, etc.) Some Bricknet systems require dialing a telephone number

via a modem to first establish a link before communicating. This is also setup in the Network Port configuration. You can also select the defaults and change the Network Port configuration later.



See *The ScadaBuilder Hierarchy* (on page 73).



To create a Bricknet session, select the **Setup | Network Sessions...** menu. A dialogue window will pop up for naming the session. Accept the default name or enter a new one. If you already have at least one Network Session, you can also simply click on the “New” button on the right-hand side of the Network Session window.

Network Port - After selecting the protocol, choose the Network Port. ScadaBuilder only displays the names of ports with configurations that are compatible with the selected protocol. If you don’t see the port that you want listed, go back and check the Network Port configuration.

Network Address - A Bricknet node must be assigned an address between 0 and 65535. You also have the option of using DIP switches on the Controller (EtherLogic or ScadaFlex Plus family) or a register for changing the slave address. This is accomplished through fields under the “Misc” Tab. If “Register” is chosen, you must select the tag name of an integer register and configure it as “retained” (nonvolatile).

Timing:

Rcv character timeout:
5000 (ms)

Event gap:
0 (ms)

Session gap:
0 (ms)

Probe interval:
0 (sec) ☐ Disable

Connect timeout:
35 (sec)

Session activity timeout:
30 (sec)

Let's take a look at the Timing Tab. In Bricknet, the probing system works virtually the same as other transactional protocols with the exception of probe mode. When a unit (by Network Destination) is placed into communications failure, instead of retrying with message at the probe interval, Bricknet will retry simply with a Probe command (like a ping for TCP/IP). This reduces the number of bytes that go out over the network trying to contact a unit that is already in failure leaving more network bandwidth for working units.

Creating a Bricknet Destination



See *The ScadaBuilder Hierarchy* (on page 73).

Project Manager - D:\proj\Help\Bricknet\Bricknet System.sbp

Nodes Files

Bricknet System

Central

Node1

Node2

Node Settings

General Target Configuration Advanced

Node properties:

Node name: Node1

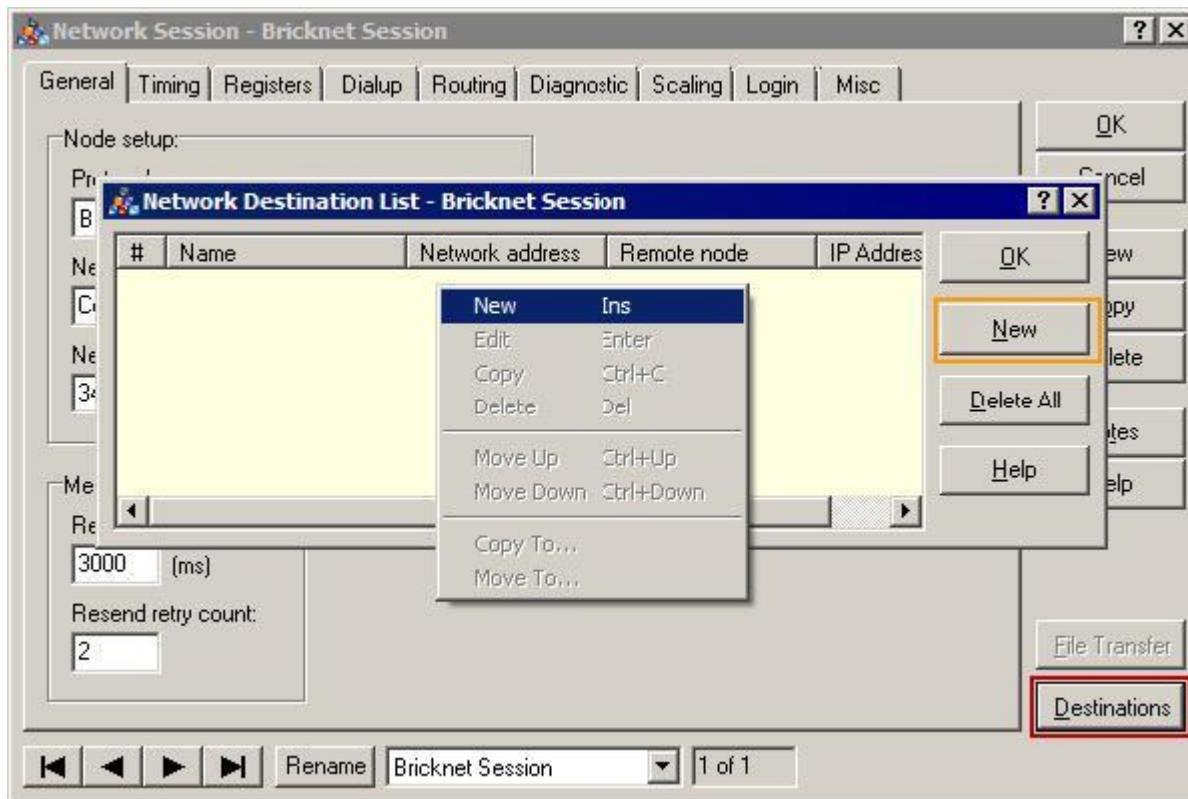
Node address: 123 (default)

Controller/RTU model: Lassen Controller

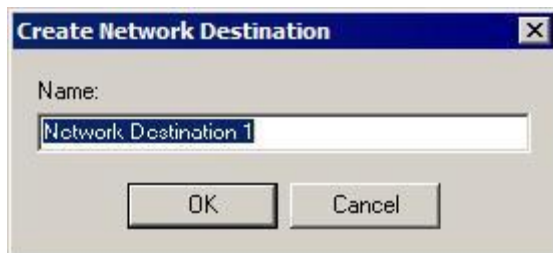
Controller/RTU options:

To utilize Bricknet, all nodes in a Bricknet system must be configured in the current ScadaBuilder Project. Each node should have its default address in the Node | Settings | General tab set up so the Destination configuration can get its address easily. If this is not done, the node of interest can still be selected but the address of the remote's Bricknet Network Session has to be entered manually.

Each remote node that the Bricknet Network Session needs to talk to must be setup as a Network Destination. To create a Network Destination, Open the Bricknet Session and click on the Destinations button in the lower right hand corner to bring up the Network Destination List.



Right click in the list area or click on the New button to start a new Network Destination. You should see this dialog:



Use the default name or give it a new one. You can name it anything you like. If you do use a name it is recommended that the Site name be used and the remote address be included so the Destination is readily identified.

Click Ok .

Select the node of interest and if needed enter the Network Address of that unit's Bricknet Session or configuration.

Network Destination - Node1

Setup | Routing | Dialing

Unit destination:

Network address: 123

Com fail map: (none)

Remote node: Node1

Com disable map: (none)

Central
Node1
Node2

Com statistics:

Select Register: Map

Statistic	Register
-----------	----------

OK
Cancel
New
Copy
Delete
Notes
Help
Events

Navigation buttons: Previous, Next, First, Last, Rename, Node1, 1 of 1

Optionally, you can also map the Com Fail flag to a boolean registers to monitor the general health of the slave. You can also map the Com Disable boolean and communications statistics for more control and monitoring capability. See *Address Com Statistics* (see "Com Statistics" on page 247) for more information.

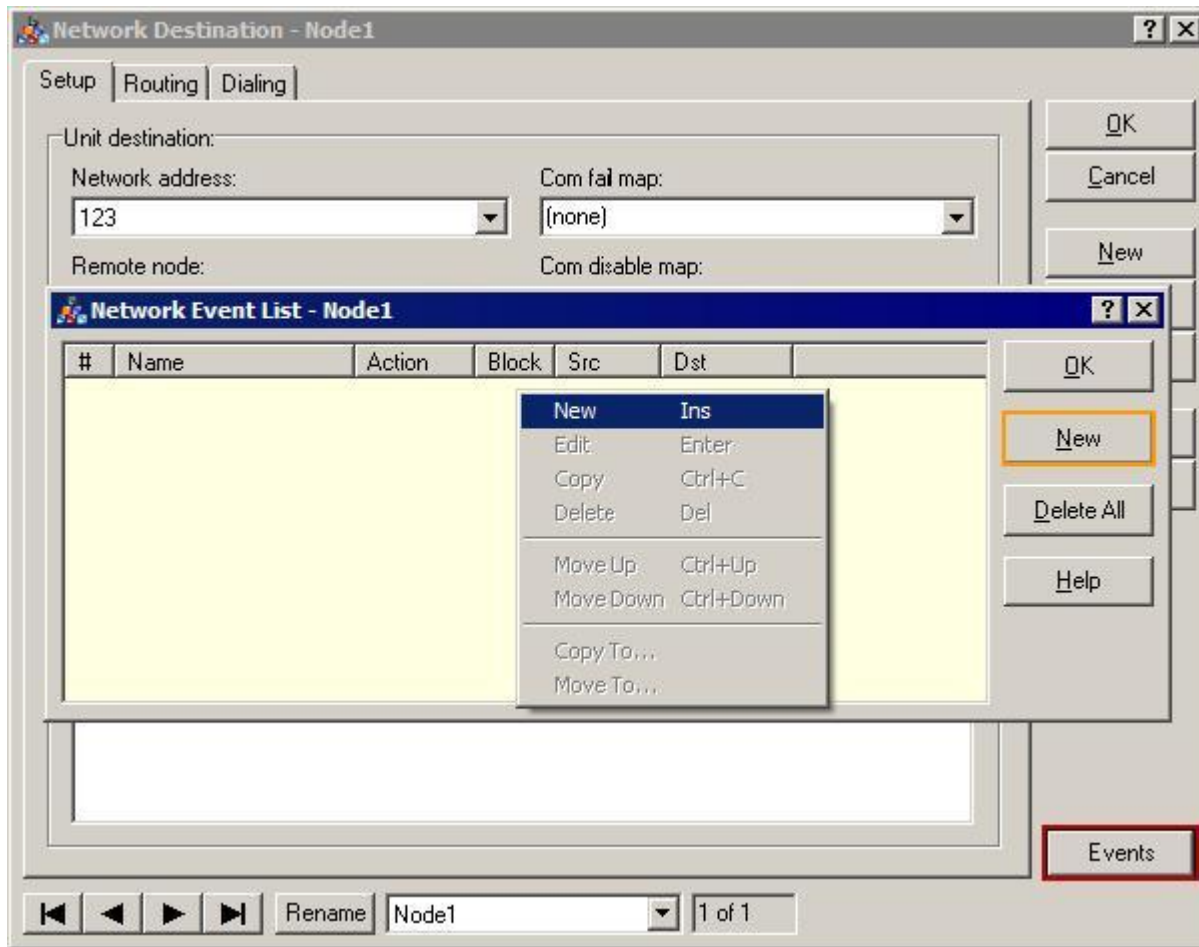
Creating Bricknet Network Events



See *The ScadaBuilder Hierarchy* (on page 73).

Once the basic Network Session and Network Destination have been set up, you're ready to set up the events for sending messages. Click on the "Events" button in the lower right-hand corner of the Network Destination window. This will bring up a "Network Event List" window.

To create an event, click on "New" or right click in the list area and select New.



Event name (optional):
Read UI's

Event message:
Action: Read
Source: AO1 (20501) Index: (remote side)
Destination: UIA1 (20001) Index: (local side)
Block size (or select last register of block): 2 (destination register block)

(Integers)	(Integers)
20501 (AO1)	20001 (UIA1)
20502 (AO2)	20002 (UIA2)

The resulting Network Event window has an optional field for the event name. Naming your Network Events makes it easier to understand what the events do in the future.

First we need to define the type of data transfer. An event can read registers, write registers, or probe a remote node. Fill in the remote nodes address, and “Action” (read, write or probe) in the appropriate fields.

The Source and Destination registers are read from each node involved and show up in the selection list. There is no other configuration for the register as the data type and existence of the registers is already known from the ScadaBuilder database.

For efficiency, all data transfers are done in blocks of multiple registers. Enter the number of registers to be transferred as the “Block size”.

For a “read” event, data will be transferred from the remote node to matching registers in the local controller. Select the Source data type and Index (register number) in the remote slave, and the FIRST register of the block that the data is to be transferred to in the controller.

For a “write” event, the Source will be the first register of a block of registers in the local controller and the destination will be the corresponding first register in the remote slave that the data will be transferred to. See **Network Events Reference** (on page 255) for more details.

Click on the Activation tab to setup WHEN this event fires.

See **Network Event Activation** (on page 259) section for more details.

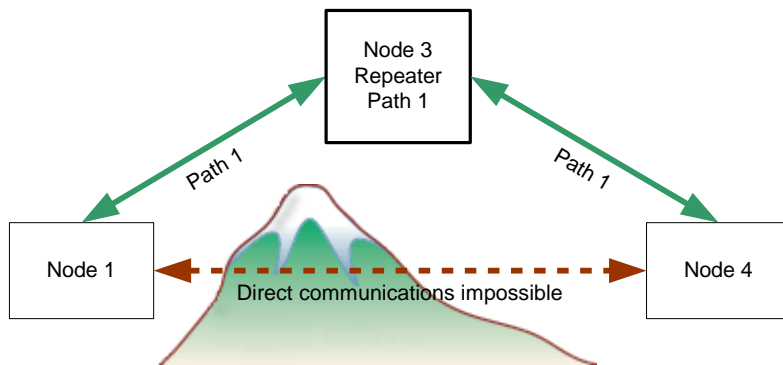
Synchronizing The Real Time Clock over Bricknet

This is done the same way as over Modbus Master and Slave Interfaces with the RTCGet and RTCSet Action options. Please see **Real Time Clock Network Events** (on page 267) for more details.

Understanding Bricknet Routing

Bricknet is a much more sophisticated protocol than our standard Modbus. It has many features that allow it to be used in much more complex ways than the typical master and slave relationship.

One of those features is Routing--commonly referred to as Store and Forward.

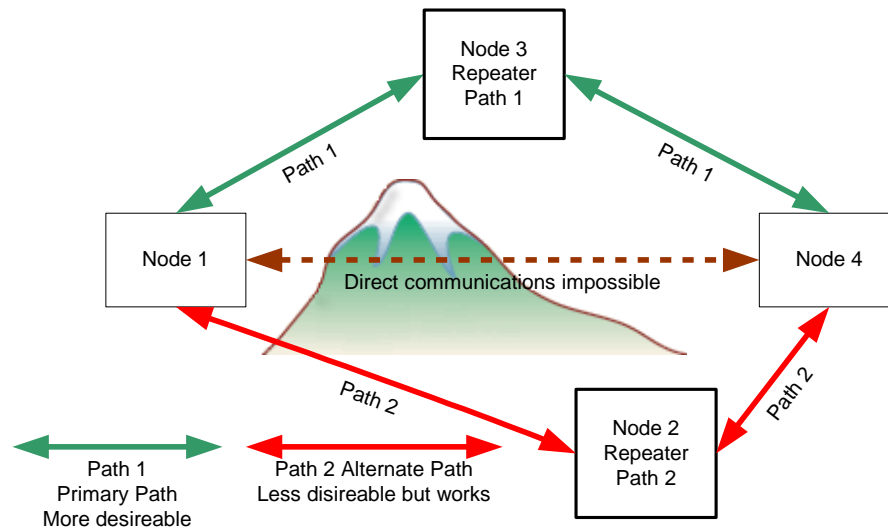


Here we have an example of a small system. Clearly the mountain is in the way of talking from Node 1 to Node 4. Node 1 is trying to talk with Node 3 (no problem) and Node 4. Node 4 clearly has a mountain in the way preventing radio coverage and therefore communication with Node 1 is impossible.

Node 3 has clear radio coverage to both units and would make an ideal repeater.

Routing with Bricknet has some other features that can be utilized in case Path 1 goes down.

Consider this diagram. Path 2 is a viable communication medium to the distance to Node 2 is not as good as Node 3, it still can work with some reliability.

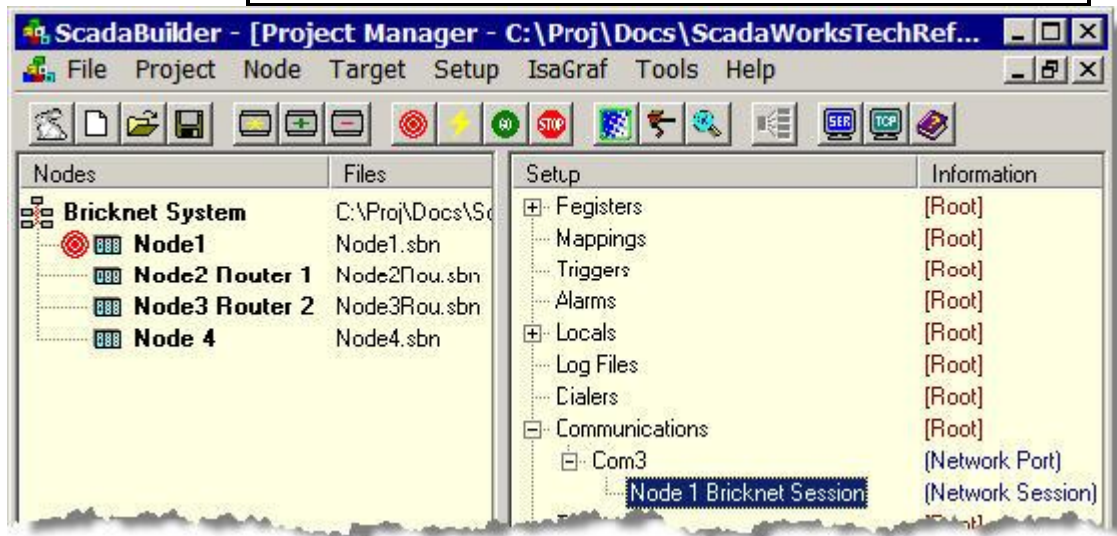
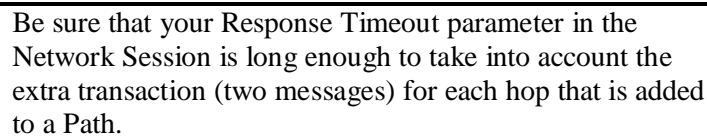


With Bricknet, when a primary path goes down, a second, third or more paths can take over for the downed link. The initiating controller continues to try the primary link. When it gets a response from the primary link, the initiating controller will switch back to that path.

Setting Up a Bricknet Routing System

When setting up a Bricknet system of any kind, it is necessary to define all nodes within one project. You can configure up to 255 nodes in a single project (this is currently the limit to an ISaGRAF group not a limit of the protocol(s)).

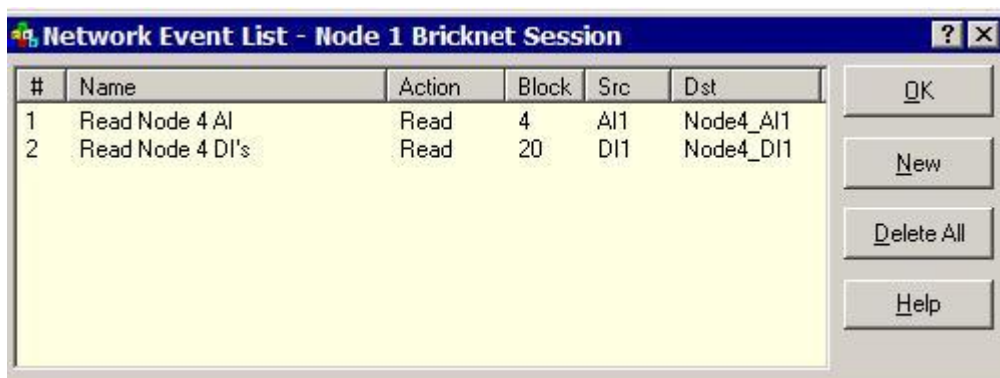
Here we are setting up Node1, Node 2 Router 1, Node 3 Router 2 and Node 4. Each of the units shown in the previous section are here and must have a Bricknet session created with an individual address for the network; see *Creating a Bricknet Session* (see "Creating a Bricknet Interface" on page 348) section.



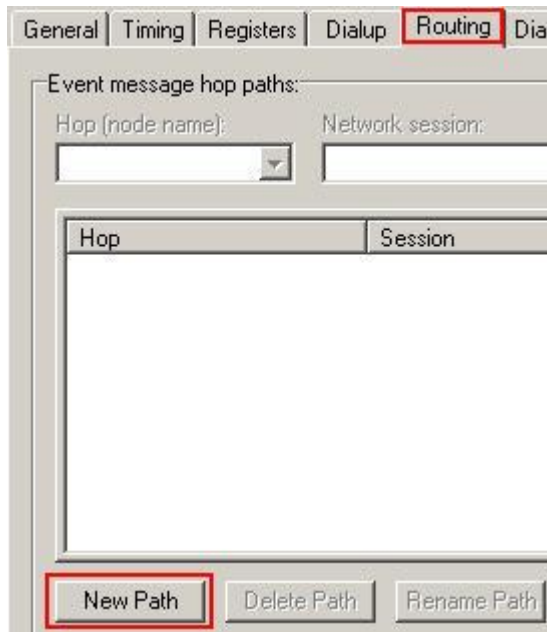
In the case of this system the addresses and the session names are:

Node Name	Address	Session Name
Node1	1	Node 1 Bricknet Session
Node 2 Router 1	2	Node 2 Bricknet Session
Node 3 Router 2	3	Node 3 Bricknet Session
Node 4	4	Node 4 Bricknet Session

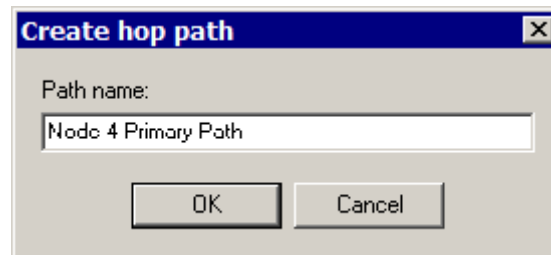
To begin to configure our system Routing, we must begin at the initiating session and create some Network Events that need to be routed. For this example we will use the following Network Events...



We can talk to Node 2 and Node 3 but for the purposes of illustration, those were purposely left out.



Now click on the Routing Tab in the Bricknet Network Session and Click on the New Path button to begin entering routes. Enter a new path name and click OK.

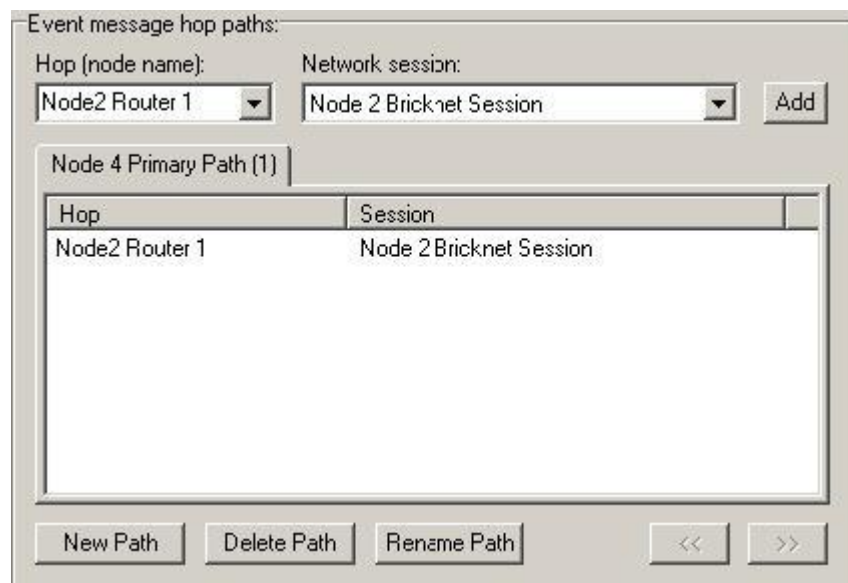


Configuring the Primary Path

This is the configuration for Path 1 and shown next to the name in parentheses.

Select the "Node2 Router 1" for the Hop node name.

From that unit select the Network Session used on the routing hop (there can be more than one Bricknet session on a unit.)



Add the Path to the list. You should see a configuration like the one to the right here.

At this point, you can download your configuration to Node 1, 2, and 4 and your Network Events will be routed through Node 2.

Configuring an Alternate Path

Event message hop paths:

Hop (node name): Network session:

Node 4 Primary Path (1) Node 4 Alternate Path (2)

Hop	Session
Node3 Router 2	Node 3 Bricknet Session

Click on the New Path button to enter more routes. Enter a new path name and click OK. This time we will call the Route "Node 4 Alternate Path"

Select the "Node3 Router 2" for the Hop node name.

From that unit select the Network Session used on the routing hop (there can be more than one Bricknet session on a unit.)

Add the Path to the list. You should see a configuration like the one to the left here.

You must now download Nodes 1, 2, and 3 for this configuration to take effect. If you do not download the routing controllers, your routes won't work and your communications will be broken.



Don't forget, download all units involved in the route after configuring that route.

Now that we have two paths, the first path or tab is the primary path and the second tab is the alternate path.

If communications goes bad on the first path, the Network Sessions will switch to the second path.

if communication is restored to the primary path, then the Network Session will stop communications on the Alternative part.

The Two or More Hop Paths

You can also add more than one hop to a path to increase distance in a system.

Hop	Session
Node2 Router 1	Node 2 Bricknet Session
Node3 Router 2	Node 3 Bricknet Session

This configuration will first send the message to Node 2 then Node 2 will repeat the message to Node 3. Node 4 will receive the message from Node 3 and respond.

Direct Paths and Hop Paths

Direct (1) Alternate Path (2)	
Hop	Session

Direct (1) Alternate Path (2)	
Hop	Session
Node2 Router 1	Node 2 Bricknet Session

You can create a Direct path by not adding any hops at all to the Path Tab. This is useful if you think a direct path might be flaky for radio coverage even though it seems to work most of the time.

When the Direct path goes down, the Alternate path takes over.

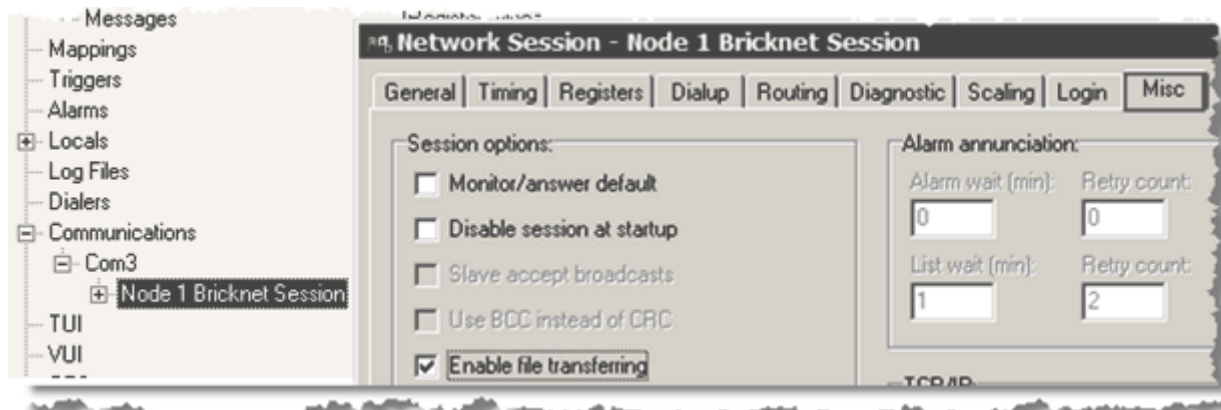
This way, the Direct path is used when it is working and is more efficient than a route path. Only when there are problems will the Alternate Path take over.

The Network Session will automatically go back to the Direct path when it goes back into service (a probe message must be answered for this to happen.)

File Transferring Over Modbus And Bricknet

To transfer files serially over a Modbus or Bricknet network requires that the Sessions for both sides (the Node or Master initiating the File Transfer Event and the Node or Slave responding) have File Transferring enabled in their respective Network Session.

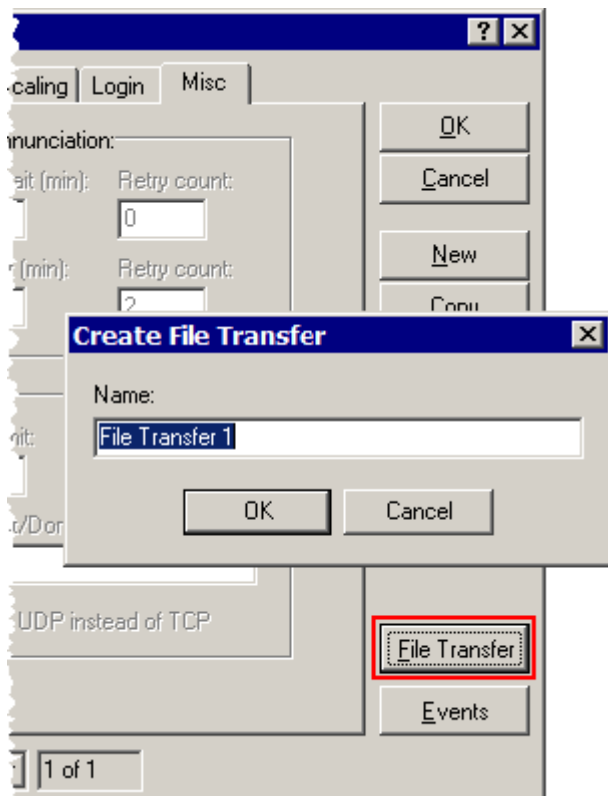
To do this, open the particular session with which you want to transfer files, click on the Misc (Miscellaneous) tab and check the Enable File Transferring box:



This must be done for every session that uses the File Transfer feature (including Modbus Slaves).



File Transferring takes some memory resources. Most of the time those memory resources will not be in use.



You're ready to configure the names of files to be transferred and the initiating Triggers and other parameters related to the data transfer.

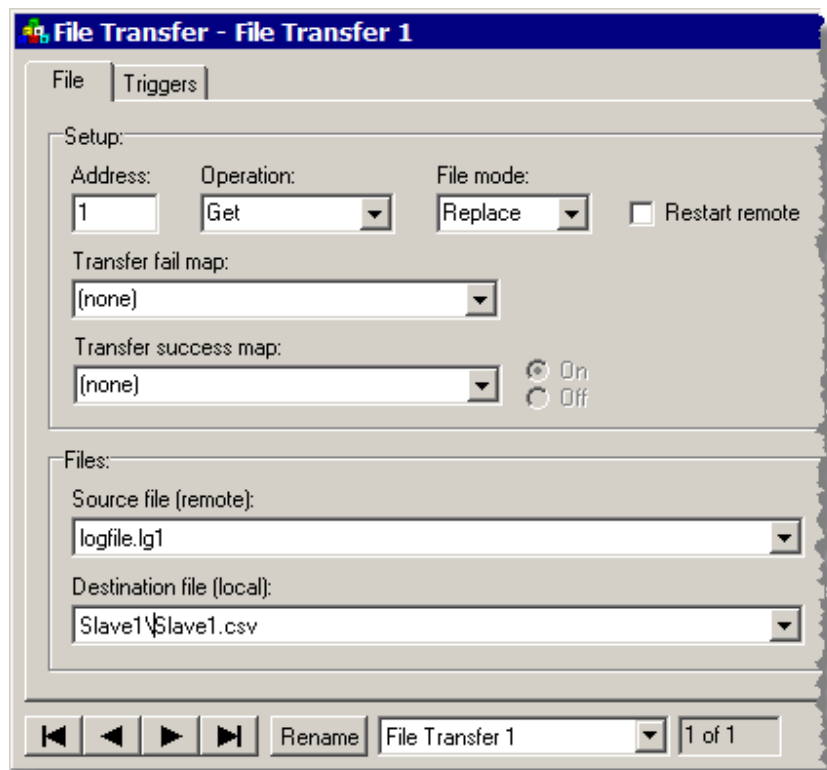
To create a File Transfer Event (Master or Bricknet), click on the File Transfer button in the lower right hand corner (after enabling file transferring.)

Operation - Specifies the transfer operation to be performed. Selecting 'Get' will retrieve a file from the server. Selecting 'Send' will send a file to the server. Selecting 'Send/Delete' will send a file to the server and then delete it from the local controller once delivery is confirmed.

Mode - Allow the programmer to configure whether the transfer will replace an already existing file or append to it.

If we are getting a file, the Destination field is the local disk on the controller. If we are sending a file, then the destination is on the remote node.

Restart Remote - check this if this File Transfer Event is downloading a new program to a controller. After the new program is downloaded successfully, the remote system will then reboot.



Triggers Tab - Any trigger can fire a File Transfer. See Using Triggers for more information.

For further information, see the following sections in this manual:

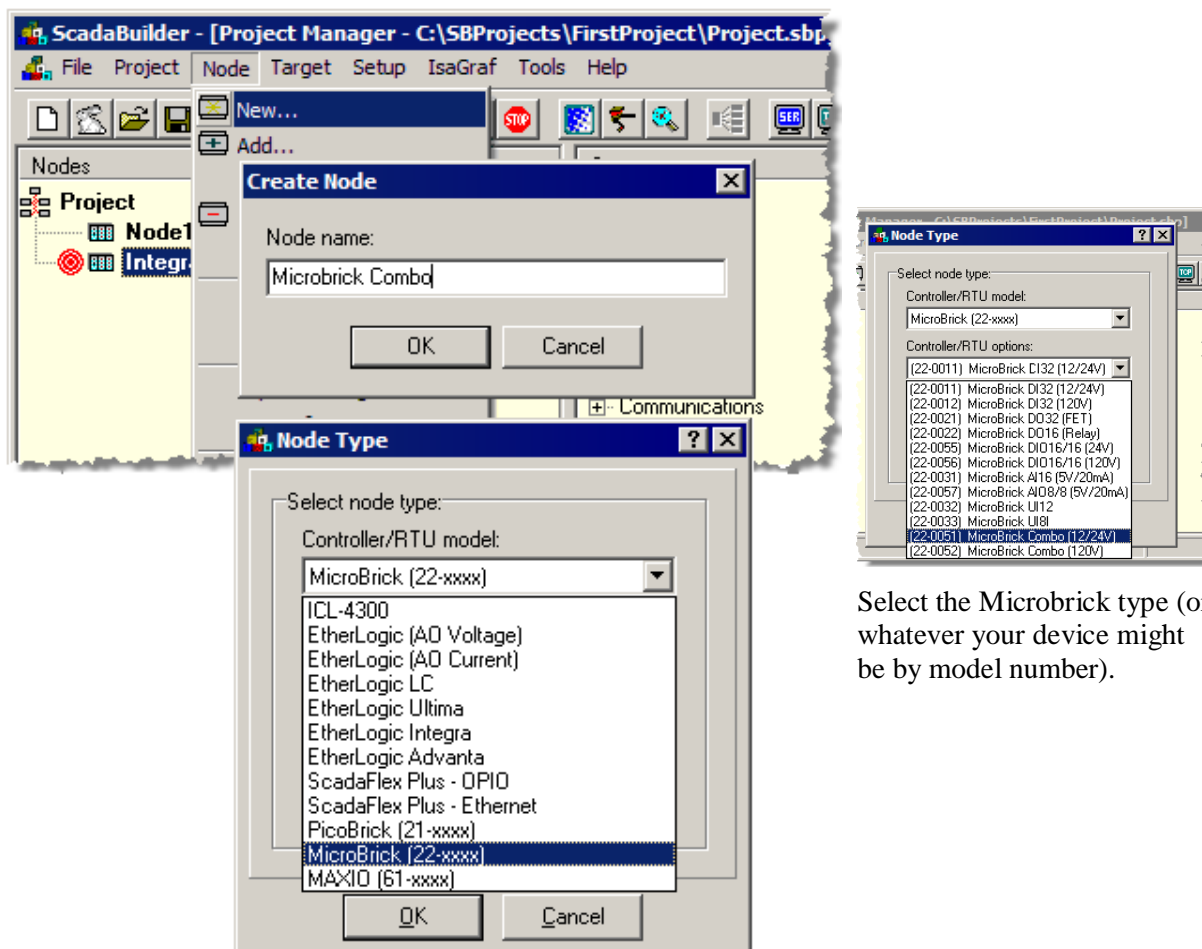
Network Session - File Transfer (see "File Transfer" on page 234), **Remote Host File Transfers** (on page 281) and **File Transfer Event Reference** (on page 279)

Using Bricknet To Communicate With I/O Modules

All I/O Expansion modules available from ICL talk ICL's Bricknet Protocol. ScadaBuilder (ICL's controller programming software) is aware of the register map of every I/O module and can access those registers directly by name and block.

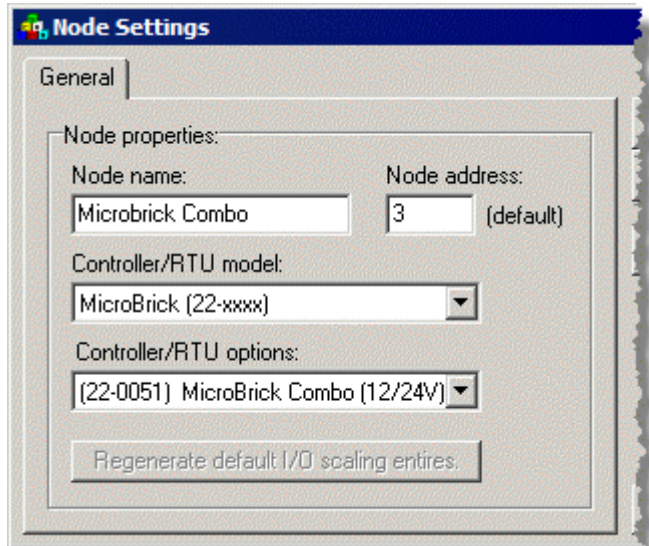
To utilize this feature, you first have to create a new Node in the ScadaBuilder project where you want to use the I/O module. We will use a Microbrick Combo module for this exercise but the concepts are the same for all other *Picobricks*, *Microbricks*, *MAXIO's* and *ScadaFlex RTU's*.

To create a Microbrick Combo Node, select the Node | New menu. Enter a name and click OK.



Select the Microbrick type (or whatever your device might be by model number).

Enter a Node address that is something other than what you main controller's address is going to be. See ***Creating a Bricknet Session*** (see "Creating a Bricknet Interface" on page 348) in the Scadaworks Technical Reference Manual for details.



You should have a project that looks like the following:



Create a Bricknet Network Session. See ***Creating a Bricknet Session*** (see "Creating a Bricknet Interface" on page 348) in the Scadaworks Technical Reference Manual for details.

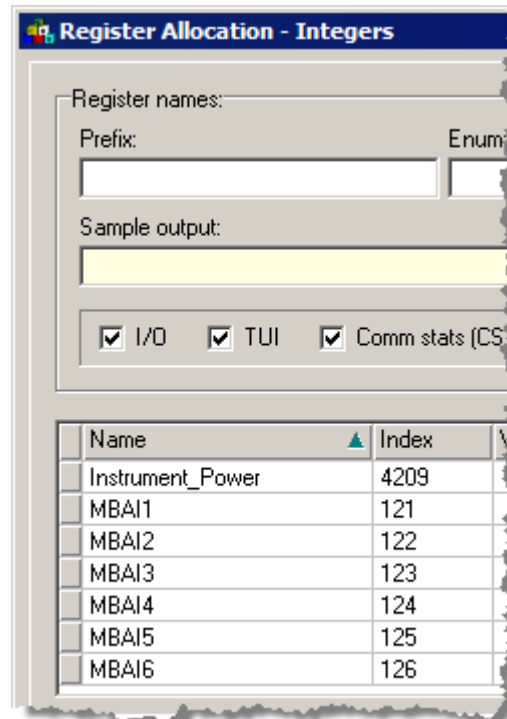
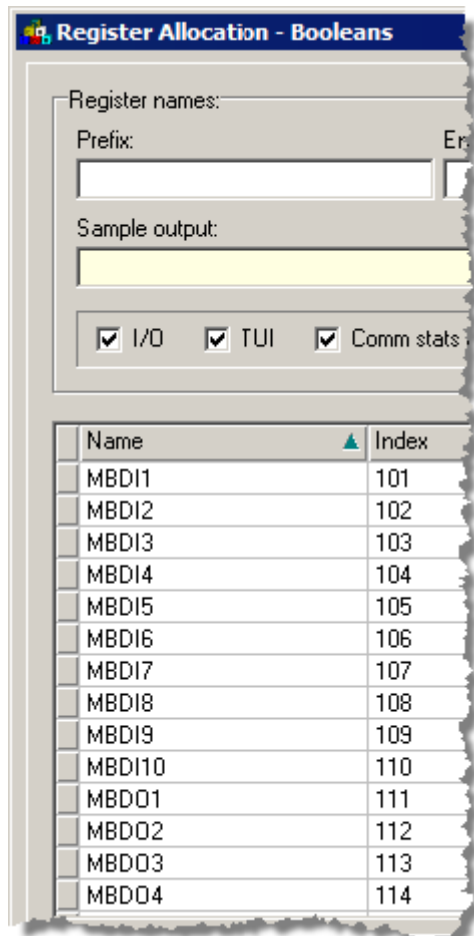
You must create registers to store locally the I/O points of the combo module. Declare the following points in the Registers section of the Setup window of ScadaBuilder. See the ***Registers*** (on page 103) section of the Scadaworks Technical Reference Manual for details.

Create the following registers to store the values into:

Booleans	Booleans	Integers
MBDI1 101	MBDO1 111	MBAI1 121
MBDI2 102	MBDO2 112	MBAI2 122
MBDI3 103	MBDO3 113	MBAI3 123
MBDI4 104	MBDO4 114	MBAI4 124
MBDI5 105		MBAI5 125
MBDI6 106		MBAI6 126
MBDI7 107		
MBDI8 108		
MBDI9 109		
MBDI10 110		

Your boolean register list should look like this.

And your integer register list should look like this:



Next we need to create the Network Destination. See *Creating a Bricknet Destination* (on page 350) for details. Select the Microbrick Combo as the remote node:

Setup | Routing | Dialing

Unit destination:

Network address:

Com fail map:

Remote node:

Controller

Com statistics:

Select

Statistic	Register

Next we need to create the Network Event. See *Creating Bricknet Network Events* (on page 352) for details.

Event	Activation	Options																						
<p>Event name:</p> <p>Event name (optional):</p> <p>MB_DI_Read_Event</p>																								
<p>Event message:</p> <p>Action:</p> <p>Read</p> <p>Source:</p> <p>DI1 (1)</p> <p>Index:</p> <p>(remote side)</p> <p>Destination:</p> <p>MBDI1 (101)</p> <p>Index:</p> <p>(local side)</p> <p>Block size (or select last register of block):</p> <p>14</p> <p>(destination register block)</p>																								
<table border="1"> <thead> <tr> <th>(DI Registers)</th> <th>(Booleans)</th> </tr> </thead> <tbody> <tr><td>1 (DI1)</td><td>101 (MBDI1)</td></tr> <tr><td>2 (DI2)</td><td>102 (MBDI2)</td></tr> <tr><td>3 (DI3)</td><td>103 (MBDI3)</td></tr> <tr><td>4 (DI4)</td><td>104 (MBDI4)</td></tr> <tr><td>5 (DI5)</td><td>105 (MBDI5)</td></tr> <tr><td>6 (DI6)</td><td>106 (MBDI6)</td></tr> <tr><td>7 (DI7)</td><td>107 (MBDI7)</td></tr> <tr><td>8 (DI8)</td><td>108 (MBDI8)</td></tr> <tr><td>9 (DI9)</td><td>109 (MBDI9)</td></tr> <tr><td>10 (DI10)</td><td>110 (MBDI10)</td></tr> </tbody> </table>			(DI Registers)	(Booleans)	1 (DI1)	101 (MBDI1)	2 (DI2)	102 (MBDI2)	3 (DI3)	103 (MBDI3)	4 (DI4)	104 (MBDI4)	5 (DI5)	105 (MBDI5)	6 (DI6)	106 (MBDI6)	7 (DI7)	107 (MBDI7)	8 (DI8)	108 (MBDI8)	9 (DI9)	109 (MBDI9)	10 (DI10)	110 (MBDI10)
(DI Registers)	(Booleans)																							
1 (DI1)	101 (MBDI1)																							
2 (DI2)	102 (MBDI2)																							
3 (DI3)	103 (MBDI3)																							
4 (DI4)	104 (MBDI4)																							
5 (DI5)	105 (MBDI5)																							
6 (DI6)	106 (MBDI6)																							
7 (DI7)	107 (MBDI7)																							
8 (DI8)	108 (MBDI8)																							
9 (DI9)	109 (MBDI9)																							
10 (DI10)	110 (MBDI10)																							

Click on the Events button in the lower right hand corner of the Bricknet Network Session you created above, This will give you the Network Event List. Click on the New button to get the following dialog. See ***Creating Bricknet Network Events*** (on page 352) in the ScadaWorks Technical Reference Manual.

Name the Event (we will be reading the DI's from the MB Combo).

Select the Action (Read),

Select the Remote Node (which is the Module you setup).

Select the Destination (this tells ScadaBuilder what data type you are going to use).

Select the source of the first DI register of the Microbrick.

Enter the block size to get all 10 DI's.

Click on the Activation tab, Enter Cyclic and 1 and click the Add button

Event | Activation | Options

Event name:
Event name (optional):
MB_DO_Write_Event

Event message:
Action:
Write

Source:
MBDO1 (111)

Index:
(local side)

Destination:
DO1 (1)

Index:
(remote side)

Block size (or select last register of block):
4 (source register block)

(Booleans)	(DO Registers)
111 (MBDO1)	1 (DO1)
112 (MBDO2)	2 (DO2)
113 (MBDO3)	3 (DO3)
114 (MBDO4)	4 (DO4)

To write to the Digital Outputs of the Microbrick Combo Module, enter the following with the same Cyclic 1 Activation:

Reading the Analog Inputs on the Microbrick is the same as reading the DI's only a different data type:

Event | Activation | Options

Event name:
Event name (optional):
MB_AI_Read_Event

Event message:
Action:
Read

Source:
AI1 (1)

Index:
(remote side)

Destination:
MBAI1 (121)

Index:
(local side)

Block size (or select last register of block):
6 (destination register block)

(AI Registers)	(Integers)
1 (AI1)	121 (MBAI1)
2 (AI2)	122 (MBAI2)
3 (AI3)	123 (MBAI3)
4 (AI4)	124 (MBAI4)
5 (AI5)	125 (MBAI5)
6 (AI6)	126 (MBAI6)

You should now have a Network Event List like the following:

Network Event List - Microbrick Combo							
#	Name	Action	Block	Src	Dst		
1	MB_DI_Read_Event	Read	14	DI1	MBDI1		
2	MB_DO_Write_Event	Write	4	MBDO1	DO1		
3	MB_AI_Read_Event	Read	6	AI1	MBAI1		

Setup Complete. Connect your I/O module to the port defined, make sure the Slave number and baud rate are correct and download the application to the controller. Consult your hardware manual for the appropriate cabling to connect your I/O module.



Different products have different I/O based on the model number and type. Interfacing to all of them is similar to what is shown here.

Numeric and Alpha Numeric Pager Sessions

Pagers and Alpha Numeric Pagers are no longer handled within Network Session / Network Alarm. These functions are handled by the Alarm / Dialer / Call group interface. These protocols are here in the Network Sessions for backwards compatibility. The Dialer interface is the preferred methodology for paging.

See *Using Alarm Dialers* (on page 169) Section for more details.

Email Protocol



See *The ScadaBuilder Hierarchy* (on page 73).

ScadaBuilder supports outgoing E-mail; both as a way of announcing events and alarms, and as a means of transferring reports, log files and other controller data using file attachments like your desktops PC E-mail.

An event in the controller can automatically dial out to an internet service provider and send an E-mail periodically, based on the time of day, based on a change in an analog level or value, or on a change in the state of a discrete boolean point. The body of the E-mail text can be freely configured to include real-time register and boolean point data, as well as historical data in the form of log files created on the internal flash disk of the controller.

An E-mail Network Session can be associated with an Ethernet port or a dialup modem port enabled for TCP/IP. In the later case, the controller will automatically dial out and establish a link to an internet provider whenever an e-mail is to be sent. The port used by the E-mail protocol may be shared by other protocols. The controller can serve as an alarm dialer as well as an E-mail client using the same modem port.



The Email interface also supports Secure Authentication for those SMTP servers that require it. If your Internet Service provider requires a password with your email account, you will need this feature.

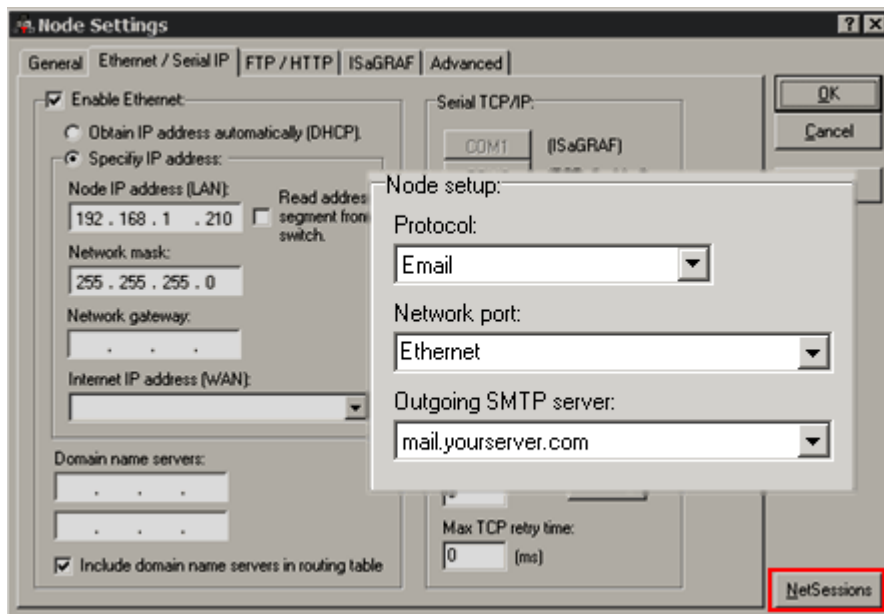


For security, many SMTP servers requires that the E-mail have a valid account in the "From" field. Otherwise the E-mail will be rejected by the server.

Creating an Email Interface

To create an e-mail session, select the **Setup | Network Sessions...** menu. Or you can double click on your Node, Select the Ethernet / Serial IP Tab and click the NetSessions button as shown here.

A dialogue window will pop up for naming the session. Accept the default name or enter a new one. If you already have at least one Network Session already, you can also simply click on the "New" button on the right-hand side of the Network Session window.



Protocol - Once you have a new Network Session window, select the E-mail protocol.

Network Port - After selecting the protocol, choose the Network Port. ScadaBuilder only displays the port names of ports whose configuration is compatible with the selected protocol, so if you don't see the port that you want named as a choice, go back and check the Network Port configuration. Remember that E-mail requires TCP/IP (under **Node | Settings | Ethernet / Serial IP** (see "Node Settings - Ethernet Tab" on page 37) tab) to be enabled.

Outgoing SMTP Server - E-mail requires this address configured in the Network Session. It is typically supplied by an Internet Service Provider (ISP). You can either enter the name of the server or the IP address into this field.

☒ Enable Ethernet:

☐ Obtain IP address automatically (DHCP).

☒ Specify IP address:

Node IP address (LAN):
 . . . ☐ Read address segment from switch.

Network mask:
 . . .

Network gateway:
 . . .

Internet IP address (WAN):

Domain name servers:
 . . .
 . . .

☒ Include domain name servers in routing table

If you choose to use a name, be sure that you have set up addresses for a Name Server in the **Node | Settings | Ethernet / Serial IP** (see "Node Settings - Ethernet Tab" on page 37) tab (a portion of which is shown at the left), so that the server name can be resolved to an IP address when the controller needs to send an E-mail.



If your E-mail is over Ethernet, then the Network Gateway must be configured to get to the Internet.

See *Node Settings - Ethernet / Serial IP Tab* (see "Node Settings - Ethernet Tab" on page 37) section for more details

If you are still confused, please see:

Appendix A, An Ethernet/Internet Primer for TCP/IP (on page 649).

On the Timing Tab of the Network Session:

Timing:

Rcv character timeout:
 (ms)

Event gap:
 (ms)

Session gap:
 (ms)

Probe interval:
 (sec) ☒ Disable

Connect timeout:
 (sec)

Session activity timeout:
 (sec)

Under the Network Session - Timing tab:

Connect Timeout (mS) - After dialing out to establish a link with the Internet Service Provider, the modem carrier must be established within this time, or the session will abort the connection.

Session Activity Timeout (mS) - If no messaging activity has occurred for this long, the session will be disconnected (modem hangs up).

On the Login Tab of the Network Session

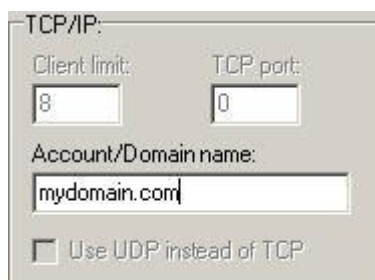
Check the Use login authentication check box if your ISP requires a login from your computer E-mail account to retrieve and send E-mails.

Username- and password- if a login is required, both fields need to be filled out. E-mail passwords are case sensitive as well.



On the Misc Tab of the Network Session

E-mails require an account or domain name that identifies the E-mail's sender. This information is set under the "Misc." tab of the network session. If a full account name is specified (such as account@mydomain.com") then the text will be inserted into the "From" field as is.



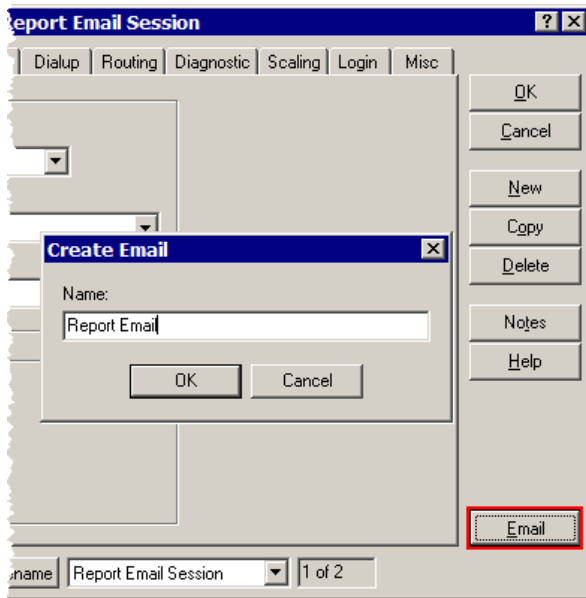
If just a domain is specified (such as "mydomain.com") then the node name will be pre-pended to the domain to create an account name. For example, if the node name is "Node1" and the domain is "mydomain.com" then the text that will be inserted into the "From" field will be Node1@mydomain.com".

Creating Emails



See *The ScadaBuilder Hierarchy* (on page 73).

Once a basic E-mail Network Session has been set up, you're ready to configure the receiver of the E-mail, the information and attached files to be included in the E-mail, as well as the Trigger(s) that cause an E-mail to be sent out.

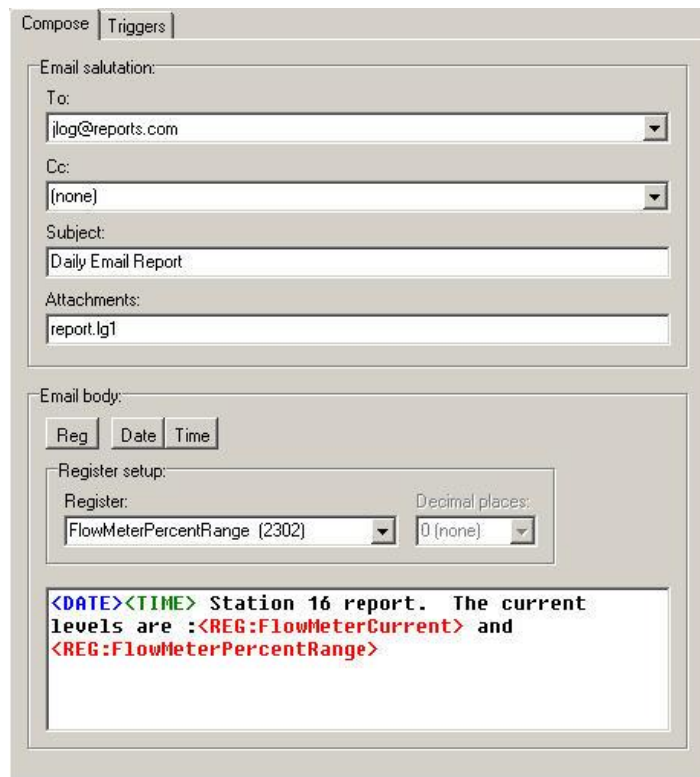


Click on the “Email” button in the lower right-hand corner of the window. A dialogue window will pop up for naming the E-mail. Accept the default name or enter a new one.

E-mail Salutation - Setting up an e-mail message in ScadaBuilder is very similar to composing an e-mail on your desktop PC. Similar fields are filled in with the destination and “cc” E-mail address(es), a subject line, and a way of linking attachments to the E-mail message. When multiple E-mail addresses or file names are used in a single field, they are separated by “,”.

Attachments - allow log files and data files of any kind to be attached to the E-mail.

E-mail Body - The body of the E-mail message can be any combination of text, real-time register or boolean data, the time and/or the date. The position of time, date and register data is indicated by bracketed names. A register is added by selecting its tag name in the Register field, then clicking on the “Register” button”. The time and date are added by simply clicking on the appropriate button.



To define your Trigger or Triggers, click on the Triggers tab.

To specify a Trigger, select a previously defined trigger in the “Trigger” window or add a [New] one, then click on the “Add” button. More than one trigger can be used for an Email. For example, the Email could be sent whenever one of the analog levels changes and once per hour.

Email Reference

E-mail provides a way to send messages to recipients over a TCP/IP network, using Ethernet or dialup. An E-mail message can be composed to include both literal text and current data register values. Date and time stamps may also be inserted. An E-mail is activated by specifying one or more external triggers.

To

This specifies the list of "To" recipients for the E-mail message. Separate multiple recipients with a single ';' (semi-colon) character.

To Buffer

Specify a buffer (message) register that contains the "To" information for the email recipient.

Cc

This specifies the list of "Cc" recipients for the email message. Separate multiple recipients with a single ';' (semi-colon) character.

Cc Buffer

Specify a buffer (message) register which contains the email address of a CC (carbon copy) recipient.

Subject

This specifies the subject line for the email message.

Attachments

This specifies the list of file attachments for the email message. Files may specified using paths and/or wild card characters. Separate multiple file entries with a single ';' (semi-colon) character.

Message Body

The body specifies the message that is sent to the recipients when the email is activated. The value of one or more registers along with the date and time can be inserted into message body.

Register

Selects the register for which the value will be inserted into the email message when the "Register" button is clicked.

Decimal Places

Specifies the number of decimal places that will be printed for a floating point register when inserted into the email message with the "Register" button.

Register Button

The "Register" button inserts the register specified in the "Register Setup" box into the email message. The current value of the register will be printed whenever the email is activated.

Date Button

The "Date" button inserts the date into the email message. The date (month/day/year) when the email was activated will be printed in the message.

Time Button

The "Time" button inserts the time into the email message. The time (hour:min:sec) when the email was activated will be printed in the message.

Trigger

This allows you to select the Triggers that cause the Email to be activated. Individually select each desired Trigger, then click the Add button.

Email Date Format

Check this box to set the date format that is inserted into the <DATE> field of an Email body.

Log Mode

This is a diagnostic option that allows you to log message interactions with the SMTP server when an Email is being transmitted. The messages are stored on the target controller in the file "EMAIL.LOG".

There are three logging options:

- Send: will only log message sent to the server.
- Response: will only messages received from the server.
- Both: will log messages sent to and received from the server.

NOTE: Due to the system resources required to do logging the overall performance of the target controller can often degrade significantly. Logging should only be used for diagnostic purposes.

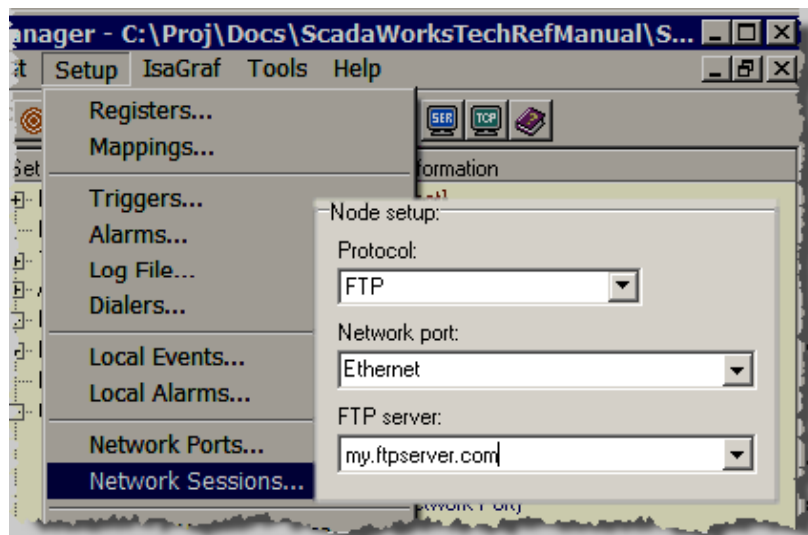
Creating an FTP Client Interface



See *The ScadaBuilder Hierarchy* (on page 73).

To create an FTP session, select the **Setup | Network Sessions...** menu. A dialogue window will pop up for naming the session. Accept the default name or enter a new one. If you already have at least one Network Session already, you can also simply click on the “New” button on the right-hand side of the Network Session window.

Protocol - Once you have a new Network Session window, select the FTP protocol.



Network Port - After selecting the protocol, choose the Network Port. ScadaBuilder only displays the port names of ports where the configuration is compatible with the selected protocol. If you don't see the port that you want named as a choice, go back and check the Network Port configuration.



Remember to use FTP, tab you must “Enable TCP/IP” under the **Node | Settings | Ethernet/Serial IP tab** (see "Node Settings - Ethernet Tab" on page 37) and Enable FTP under the **Node | Settings | FTP/HTTP tab** (see "Node Settings - FTP/HTTP Tab" on page 46).

FTP Server - This is the TCP/IP address of where your files are going to go on the Internet. You can either enter the name of the server into this field (get this name from your ISP) or the IP address.

Node Settings

General | **Ethernet / Serial IP** | FTP / HTTP | ISaGRAF

☒ Enable Ethernet:

☐ Obtain IP address automatically (DHCP).

☒ Specify IP address:

Node IP address (LAN):
 ☐ Read address segment from switch.

Network mask:

Network gateway:

Internet IP address (WAN):

Domain name servers:

☒ Include domain name servers in routing table

If you choose to use a name, be sure that you have set up addresses for a Name Server in the **Node | Settings | Ethernet / Serial IP** window (a portion of which is shown at the left), so that the server name can be resolved to an IP address when the controller needs to send an e-mail.



If your FTP Transfer is over Ethernet, then the Network Gateway must be configured to get to the Internet.

See *Node Settings - Ethernet / Serial IP tab* (see "Node Settings - Ethernet Tab" on page 37) section for more details

If you still have questions about networking, please see:

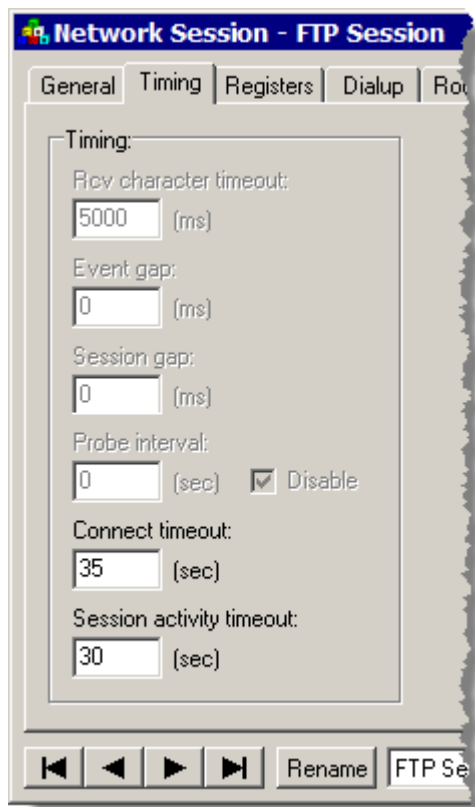
Appendix A, An Ethernet/Internet Primer for TCP/IP (on page 649).

On the Timing Tab of the Network Session:

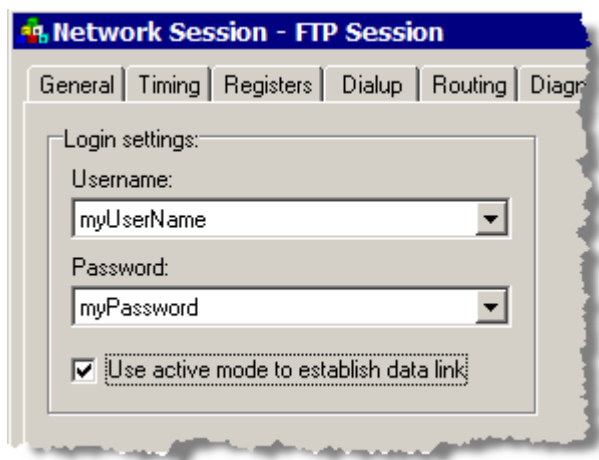
Under the Network Session - Timing tab:

Connect Timeout (mS) - After dialing out to establish a link with the Internet Service Provider, the modem carrier must be established within this time, or the session will abort the connection.

Session Activity Timeout (mS) - If no messaging activity has occurred for this long, the session will be disconnected (modem hangs up).



On the Login Tab of the Network Session



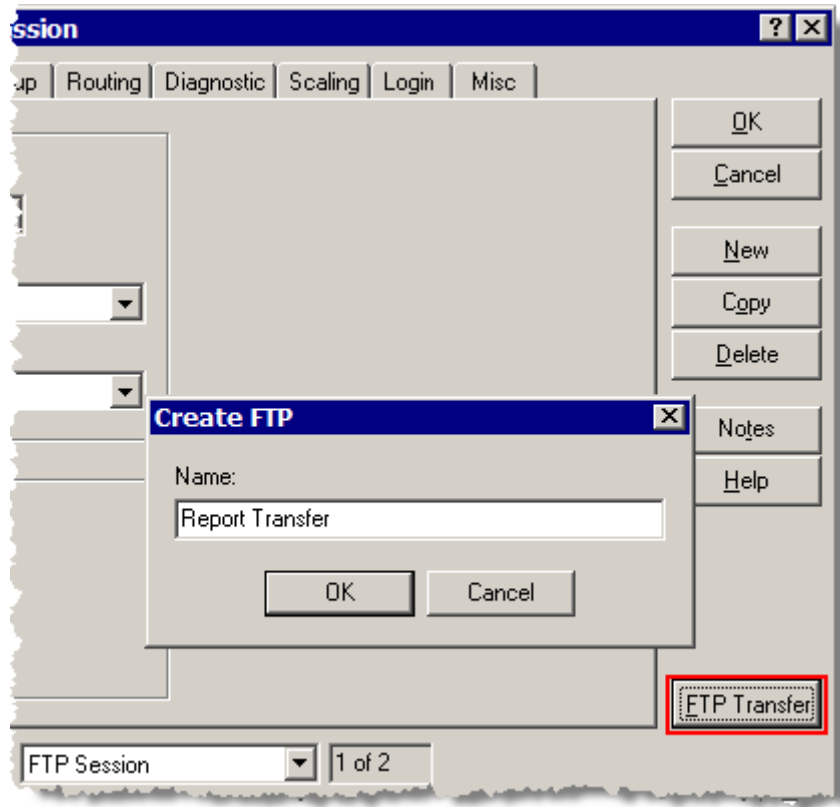
Check the Use login authentication check box if your ISP requires a login from your computer to retrieve and send files.

Username- and password- if a login is required, both fields need to be filled out. FTP passwords are case sensitive as well.

Creating an FTP Event



See *The ScadaBuilder Hierarchy* (on page 73).



Once a basic FTP Network Session has been set up, you're ready to configure the names of files to be transferred and the initiating Triggers and other parameters related to the data transfer.

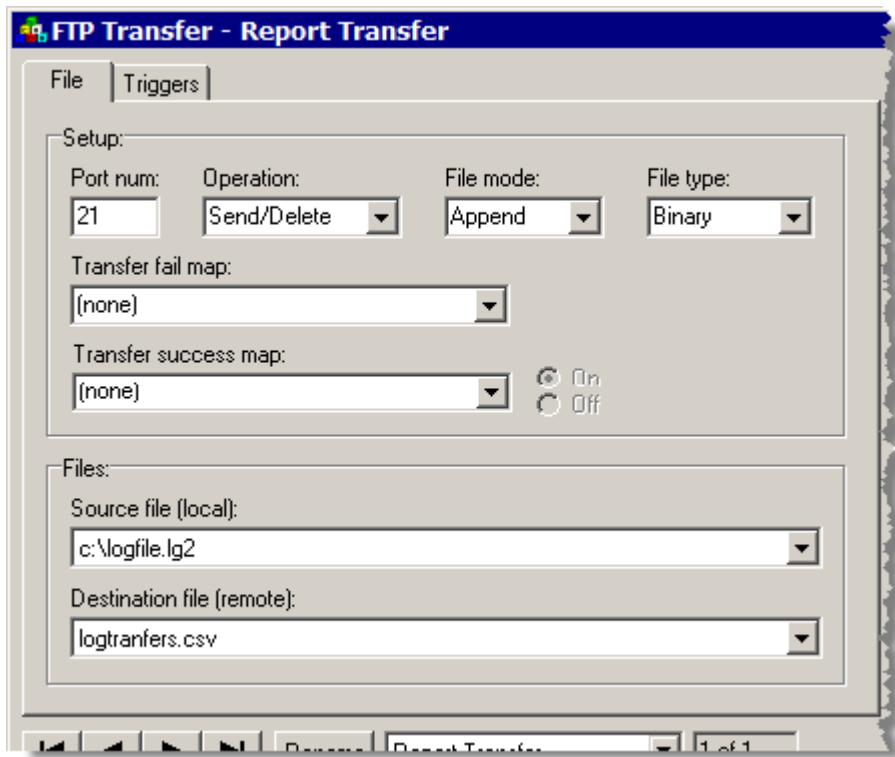
Click on the "FTP Transfer" button in the lower right-hand corner. A dialogue window will pop up to name the FTP event. Accept the default name or enter a new one.

Operation - Specifies the transfer operation to be performed. Selecting 'Get' will retrieve a file from the server.

File Configuration

Selecting "Send" will send a file to the server. Selecting "Send/Delete" will send a file to the server and then delete it from the controller after a successful transfer.

File Type - Specifies the file transfer method. The 'Binary' method treats the file as raw data without translating any characters. The 'Text' method translates a LF character to a CR/LF combination on output and translates a CR/LF combination to a LF character on input.



File Mode - This specifies the method used when sending a file to the server. Selecting 'Replace' will overwrite the file if it already exists. Selecting 'Append' will add to the end an existing file, or create a new one if a file does not already exist.

Port Num(ber) - This specifies the TCP/IP port number that the local client will use to connect to a remote FTP server (that is listening for client connections). The default port number is 21.

Source File - The name of the file to be transferred. When getting a file the source file is on the server side. When sending a file the source file is on the client side. If the file does not exist, an FTP error will be generated. When "String" is selected from the drop down tree, the file and path are entered as a literal string. When a Message (buffer) is selected, the file and path are entered into the selected message register. This allows the file/path to be changed at runtime.

In Pinnacle and later controllers, the source file will also support wildcard ie: "*.csv" or "*.log" or "*.*". When wildcards are used, the Destination file (remote) is ignored. Wildcards only work on Send commands.

Destination File - The destination file name. When getting a file, the destination file name refers to the client side. When sending a file the destination file name refers to the server side. When "String" is selected from the drop down tree, the file and path are entered as a literal string. When a Message (buffer) is selected, the file and path are entered into the selected message register. This allows the file/path to be changed at runtime.

FTP Event Reference

FTP (File Transfer Protocol) provides a way to transfer files over a TCP/IP network, using Ethernet or dialup. The local machine is an FTP client which can connect to a remote FTP server to perform a specified transaction. An FTP transfer is activated by specifying one or more external triggers.

Operation

This specifies the transfer operation that will be performed with the FTP server.

Selecting 'Get' will retrieve a file from the server.

Selecting 'Send' will send a file to the server.

Selecting 'Send/Delete' will send a file to the server and then delete it from the client.

Selecting 'Get/Delete' will get the file from the server and then delete it from the server.

File Type

This specifies the method to be used when transferring the files. The 'Binary' method treats the file as raw data without translating any characters. The 'Text' method translates a single LF character to a CR/LF combination on output, and translates a CR/LF combination to a single LF character on input.

File Mode

This specifies the method used when sending a file to the server. Selecting 'Replace' will overwrite the file if it already exists. Selecting 'Append' will add to the end of the file if it already exists (otherwise it will be created).

Port Number

This specifies the TCP/IP port number that the local client will use to connect to a remote FTP server (that is listening for client connections). The default port number is 21.

Transfer Fail Map

This specifies a Boolean register that is used to map the transfer failure status. If an FTP transfer was unable to complete (after retries) then a value of '1' will be loaded into the register, indicating a communications failure. When an FTP transfer is successful then the register will be cleared back to '0'.

Transfer Success Map

This specifies a Boolean register that is used to map the transfer success status. The associated "On/Off" control selects the value that will be written to the register when a successful FTP transfer has completed (On = 1, Off = 0). It is up to the control program to reset the value in the register to the opposing state.

Source File

This specifies the source file that will be used for the FTP transfer. When getting a file the source file is on the server side. When sending a file the source file is on the client side. If the file does not exist then an FTP error will be generated.

A hard-coded string or Message buffer may be used to specify the file name. A Message buffer allows the file name to be changed at runtime.

Pinnacle and later controllers only:

The source file will also support wildcard ie: "*.csv" or "*.log" or "*.*". When wildcards are used, the Destination file (remote) is ignored except for directory specification. The question mark '?' may also be used to specify single character wild cards. For example, a specification of '*.lg?*' will transfer all .lg1 and .lg2 files. Wildcards only work on Send commands.

Source drives may be specified by device followed by a ":" Possible devices are: "IDE:", "USB1:", "USB2:", "USB3:" and "USB4:". If the source drive is not found, then the FTP Transfer will fail.

Destination File

This specifies the destination file name that will be used for the FTP transfer. When getting a file the destination file name refers to the client side. When sending a file the source file name refers to the server side.

A hard-coded string or Message buffer may be used to specify the file name. A Message buffer allows the file name to be changed at runtime.

Pinnacle and later controllers only:

Directories may be specified with a forward slash "/" in the Destination File field. For example "logs/" will copy the file or files specified in the Source File field to the logs directory. Multiple directories may be specified such as "logs/alarmslogs/". Do not use a "/" at the beginning of the Destination File field, most FTP servers do not know what this means and the FTP Transfer will fail. DO end a directory specification with a "/" forward slash.

If a Message Register is used in the Destination File, and that Register is blank, then the source file name is used and the file is written to the root of the FTP account on the server. If a directory is specified in Destination File field then the file will be written there.

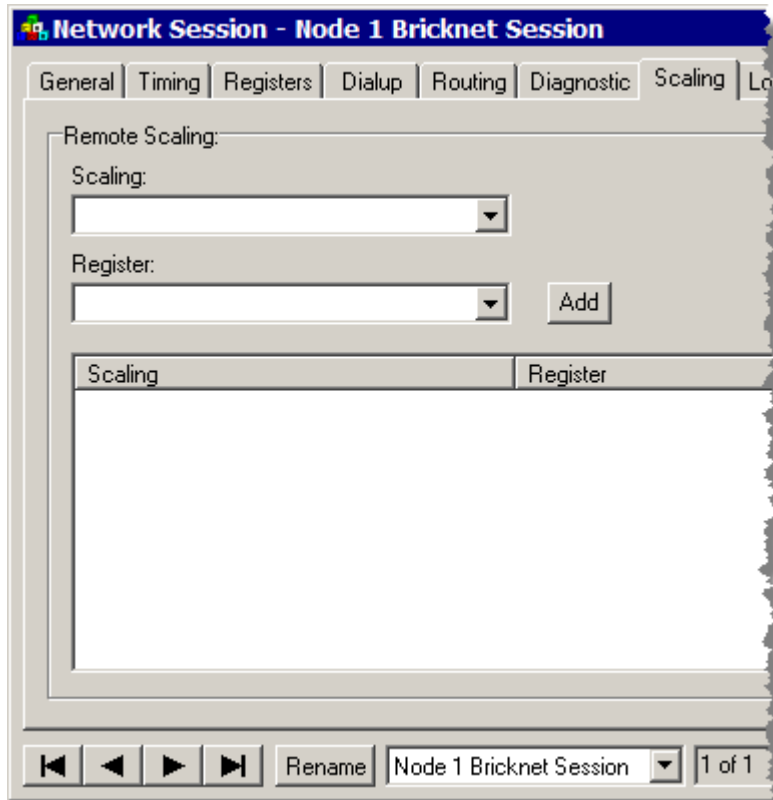
If Wildcards are used in the Source File field, then all files will be copied to the Destination File field directory. If the Destination File field is mapped to a register and the register is blank, the files will be copied to the root of the FTP account used.

All destination directories must already be created on the server. The FTP Transfer event does not create directories and the FTP Transfer will fail if the specified directory does not exist.

Trigger

This allows you to select the Triggers that cause the FTP transfer to be activated. Individually select each desired Trigger, then click the Add button.

Using Remote Scaling



Remote Scaling - With one or more remote scaling definitions, you can scale any register by simply selecting the scaling definition in the "Scaling" field and the register into which the scaled value will be written to in the "Register" field. Clicking on the "Add" button adds your selection into the Remote Scaling list for this session. To delete an entry in the list, right-click on the entry and select "Delete".

You can create a remote scale either here or in the I/O Scaling record.

To create a new one here, click the Scaling drop down box and select [New].

Creating a New Remote Scale Record

Create I/O Scale

Name:
Remote Scale 4000 - 20000 = 0 to 100

OK Cancel

0 to 100 ? x

Register scale:

Engineering units:
0 Min 100 Max

Clamp (applied to register):
0 Min 0 Max

I/O setup:

I/O mode:
(remote)

I/O range (that scale is applied to):
4000 Low 20000 High

OK
Cancel
New
Copy
Delete
Notes
Help

Navigation buttons: [Back] [Forward] [Home] [End] [Rename] Remote Scale 4000 - 2000 10 of 10

Enter a name for your scaling record. When the dialog comes up,

Enter the I/O mode as (remote) and you can choose any scaling you wish.

The I/O range is the number that will be presented to the controller (on a read event) or the controller will present to a device on a write event.

The Engineering Units are what the 4-20 represents.

Click OK.

Scaling a Remote Device's Register

Select a register to apply the scaling to and click Add.

Once configured this way, the Network Session will apply the scaling to any read, or write Network Event under a master protocol.

Any time another device writes to this register via this Network Session, the scaling will be applied.

This feature is particularly valuable when device returns A/D units or percentage instead of engineering units. This is also how you would scale I/O that came from any of ICL's RTU line--i.e. Picobrick, Microbrick, MAXIO, or ScadaFlex RTU's.

Network Session - Node 1 Bricknet Session

General | Timing | Registers | Dialup | Routing | Diagnostic | **Scaling** | Logit

Remote Scaling:

Scaling: Remote Scale 4000 - 20000 = 0 to 100

Register: Node4_AI3 (443)

Scaling	Register
Remote Scale 4000 - 20000 = 0 to 100	(Integers 441) Node4_AI1

Navigation: [Back] [Forward] [Rename] Node 1 Bricknet Session 1 of 1

Textual User Interface (TUI)

ScadaBuilder has a Text User Interface (TUI), used with operator interface terminals as well as PC computers. Without having the overhead of full graphics, the interface is simple and very efficient. With ScadaBuilder, it's extremely fast and simple to create TUI screens. Note that if a graphical interface is required, ICL's ErgoView software it is also available and takes advantage of the controller's web server capabilities.

A TUI can display virtually any kind of information in the controller. This includes integer, floating point, discrete values, logs, messages, bar graphs, etc. A TUI can also be used to change setpoints and configuration parameters with or without security.

A controller can support multiple TUI displays operating simultaneously. The TUI can be used over a serial link, over a hard-wired Ethernet or PPP dial-up connection or on the Local HMI (see Local HMI Overview) of a Pinnacle Controller. A TUI operating over a serial link works with any "dumb" terminal that supports ANSI or VT100 standard protocol. This includes PC computers with HyperTerminal emulation software that comes with Windows. A TUI operating over a hard-wired Ethernet or dialup (PPP) link requires a Telnet terminal interface. The Windows HyperTerminal program supports Telnet also.

Unlike many operator interface terminals, the configuration of displays that utilize the TUI do not require separate programming tools. Their configuration is part of the ScadaBuilder configuration so tag names are only entered once and changes to the controller program do not require separate reprogramming of the operator interface display.

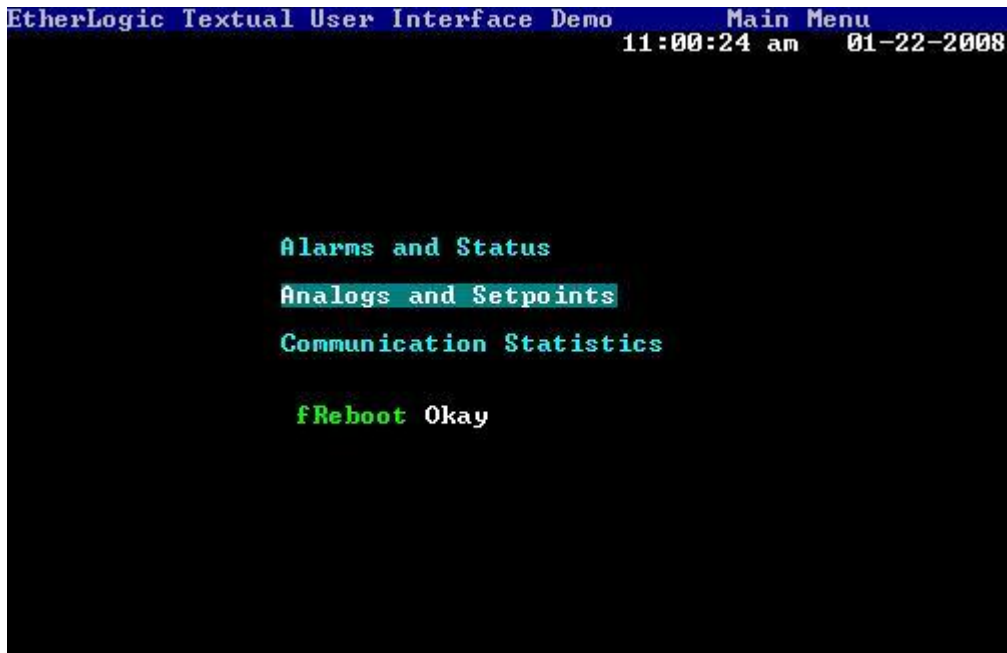


Scadabuilder has two terminal types it can use for the TUI, serial comports and telnet. Both terminal types are available in the Scadabuilder software from these buttons:

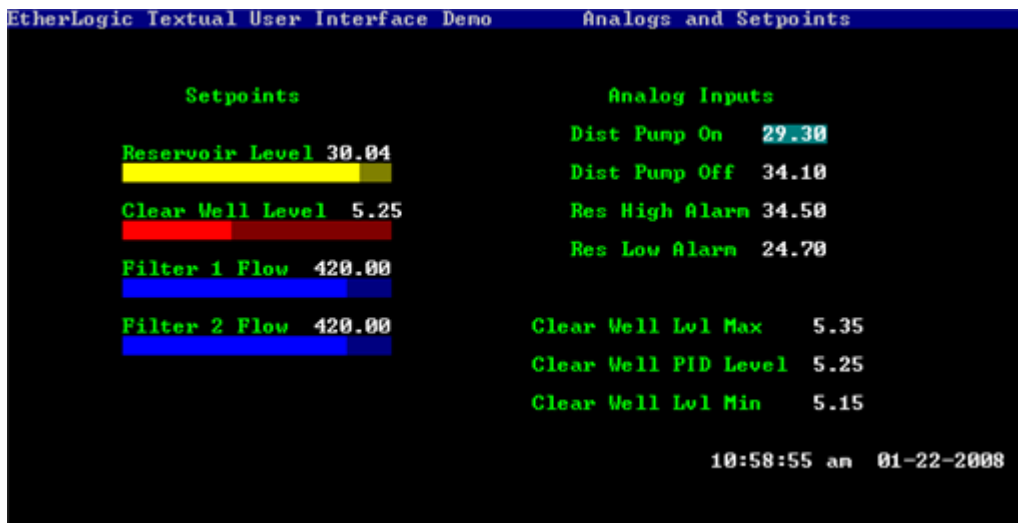


Let's look at an actual TUI interface:

A "Header" can be shared by all pages, or customized for each page. Systems with more than just a few parameters may use a main page with lower level pages in a hierarchy. The TUI has "Page Links" with selection bars. The sub-pages can also have Page Links. In small systems, the top page may be all that's required.



Integer and floating point values can be displayed as simple text and as bar graphs. The TUI designer provides extensive color and attribute controls to create attractive screens. On a TUI screen, highlighting marks a register that's "open" and ready to be changed.



"Value Lists" display text based on register values. For example, a Boolean can be "Running" or "Stopped". A numeric value can have multiple states like "Hand", "Off" and "Auto".

Note that the systems Time and Date can also be placed on any screen with multiple formats to choose from.

```

EtherLogic Textual User Interface Demo      Alarms and Status

      Pump 1 Status      Pump 2 Status      Alarms
Pump 1 Call  Call      Pump 2 Call  Call  Reservoir High  Okay
Pump 1 Run   Run       Pump 2 Run   Run   Reservoir Low   Okay
Pump 1 Auto  Auto      Pump 2 Auto  Auto
Pump 1 Fail  Okay      Pump 2 Fail  Okay   Common Alarm Alarm

      Scada Pump Inhibit Off
                                Reset

11:06:23 am  01-22-2008

```

A TUI can also display text strings and logs with scrollable windows.

These are just a few examples of some of the available features and capabilities of a TUI Text User Interface.

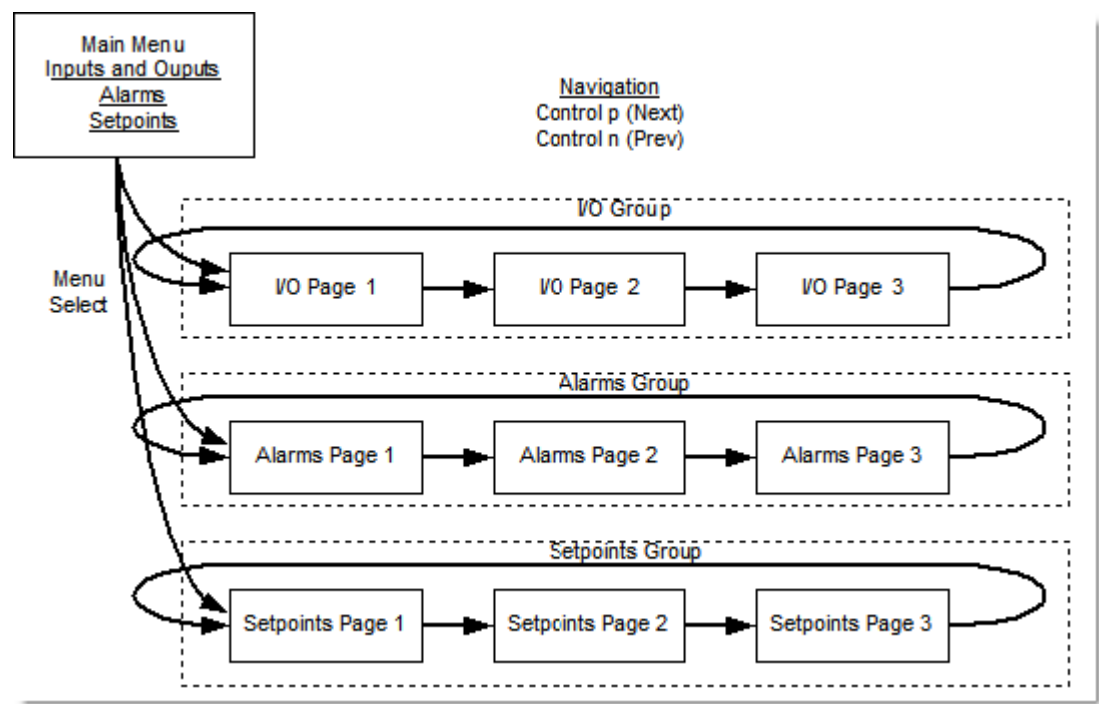
```

EtherLogic Textual User Interface Demo      Communication Statistics
Modbus Ethernet Statistics      {Access Log}
XmtCmdEnet      0 01-22-08,05:04:28 am,Noone is accessin
RcvRespEnet      0 01-22-08,05:01:16 am,Someone is access
RcvCmdEnet      2000000 01-21-08,06:49:20 am,Noone is accessin
XmtRespEnet      2000000 01-21-08,06:38:23 am,Someone is access
RcvRouteEnet      0 01-21-08,05:56:01 am,Noone is accessin
XmtRouteEnet      0 01-21-08,05:54:12 am,Someone is access
RcvTimeoutEnet    0 01-20-08,04:48:53 am,Noone is accessin
ChecksumErrEnet    0 01-20-08,04:40:20 am,Someone is access
BadContentEnet     0 01-17-08,08:01:53 am,Noone is accessin
CfgErrEnet         0 01-17-08,07:57:13 am,Someone is access
LostConnectEnet    0 01-16-08,11:01:52 am,Noone is accessin
SuccessPcntEnet    0 01-16-08,10:58:35 am,Someone is access
LastRcvCmdEnet    22011907 01-15-08,10:41:47 am,Noone is accessin
                                01-15-08,10:38:39 am,Someone is access

FTP      :/*.*: Transmit OK
ModbusTCP:Desc #66 -- TCP port released.
Http     :Normal

```

TUI's also have the capability of doing hierarchal navigation. In this example, there are three page links on a Main Menu Page. Selecting the page link gets the user into the Group. From there, the user can hit Control P (for previous) and Control N (for Next) or select the PREV and NEXT keys on a ScadaFlex Viewpoint.



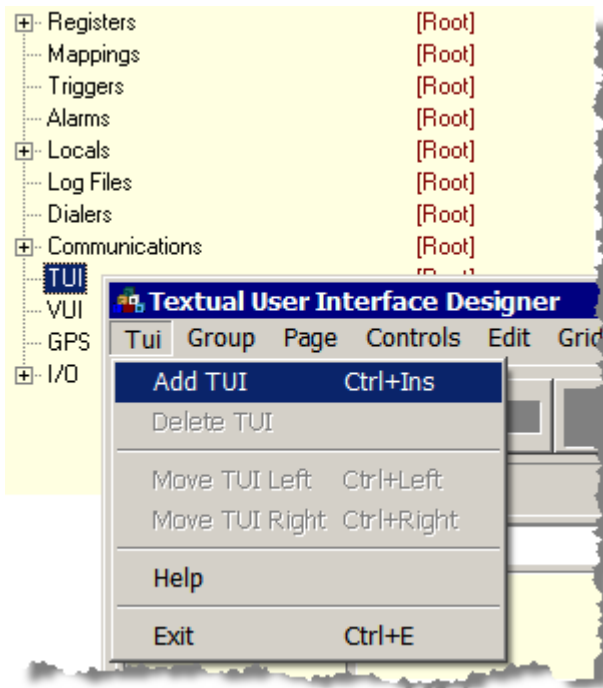
Hitting ESCape will return the user to the Main Menu Page.

The Local HMI on a Pinnacle Controller will act in exactly the same way. The difference is, no external viewer (such as a terminal emulator) is necessary.

In This Section

Creating a TUI	388
General Settings	392
Page Settings	397
Controls	399

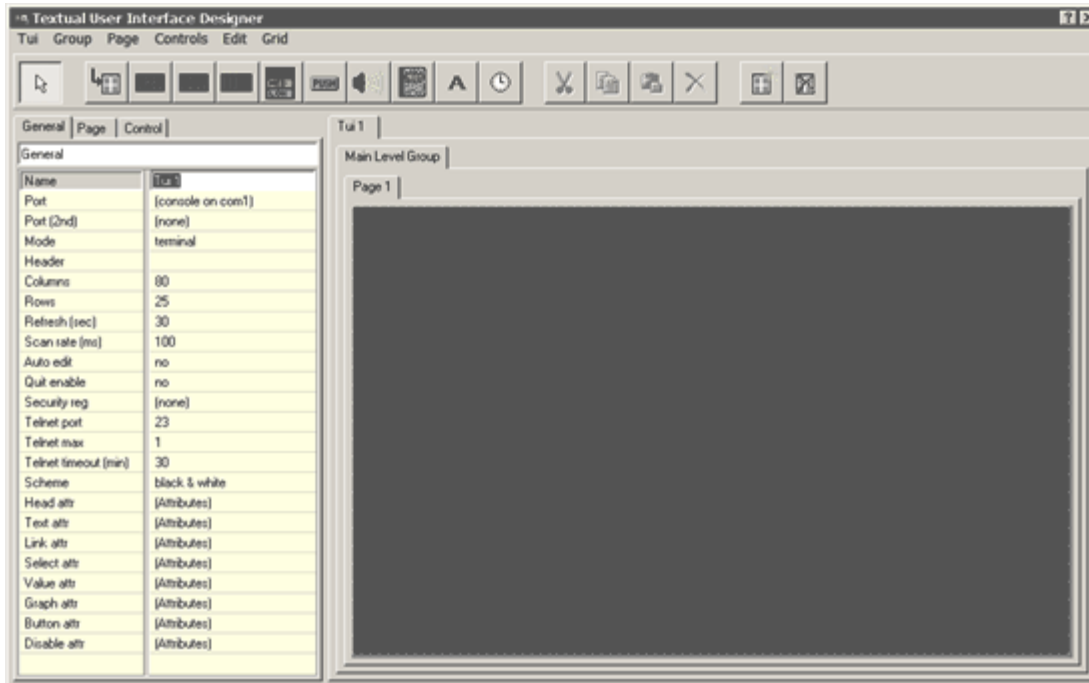
Creating a TUI



A TUI interface can be started by double clicking on "TUI" in ScadaBuilder's main Project Manager window. A blank screen is displayed. To start your new TUI, click Tui, and Add TUI.



Enter your group name and click OK. This will be the "top level" of your TUI.



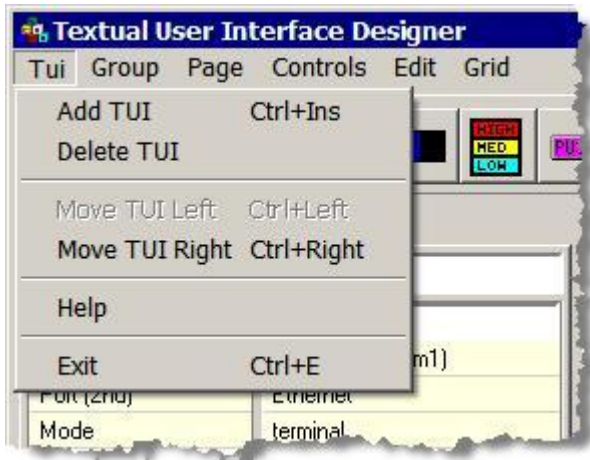
A full TUI designer window will be displayed with default parameters for a typical full PC screen (80 columns, 25 lines). These can be easily changed.

TUI Menu ItemsTUI Designer Menus

The TUI has six menu groups in the upper left-hand corner of the Design window that provide all of the tools needed to create and edit TUI pages. Some of these functions also have “hot buttons” to use them without having to go into the menu lists. These are called out wherever they exist.

TUI Menu

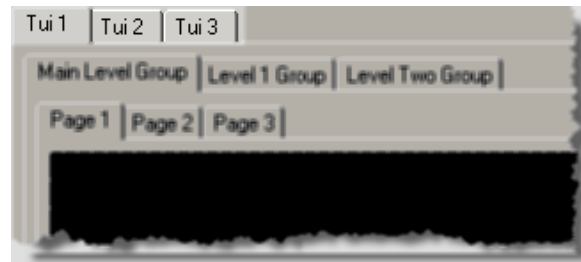
The TUI Menu has the tools that create and delete entire TUIs, as well as the tools to change the order in which they show up as tabs along the top of the TUI Design window.



Add TUI - Use this selection to create a new TUI.

Delete TUI - Use this selection to delete the currently selected TUI.

Move Left and Right - In a multiple TUI system, use these selections to change the order of the tabs along the top of the TUI Design window. The tab of the currently selected TUI is moved left or right relative to its neighboring tabs.

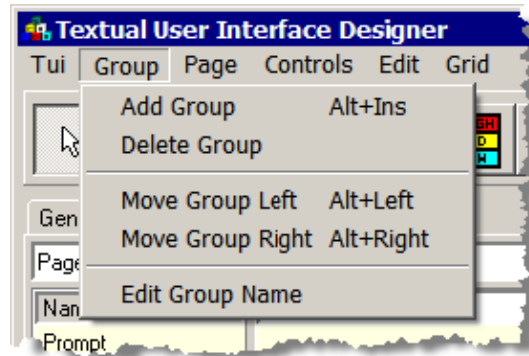
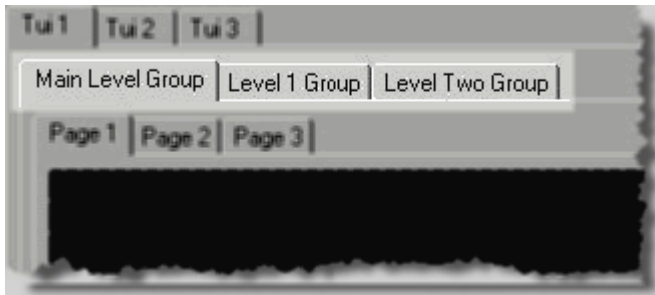


Groups

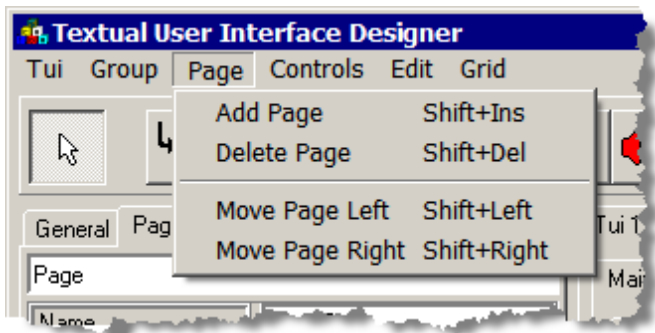
Groups allow you to define different levels in your TUI. A page link allows you into the group and Control P (or Next) and Control N (Prev) allow you navigate pages in that group. Groups are especially handy for building a TUI interface for ScadaFlex Viewpoint Displays

Group Menu

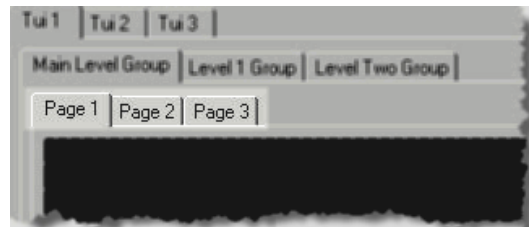
The Group menu has the tools that create and delete Groups in a TUI, as well as the tools to change the order in which they show up as tabs along the top of the TUI Design window in multiple Group TUIs.



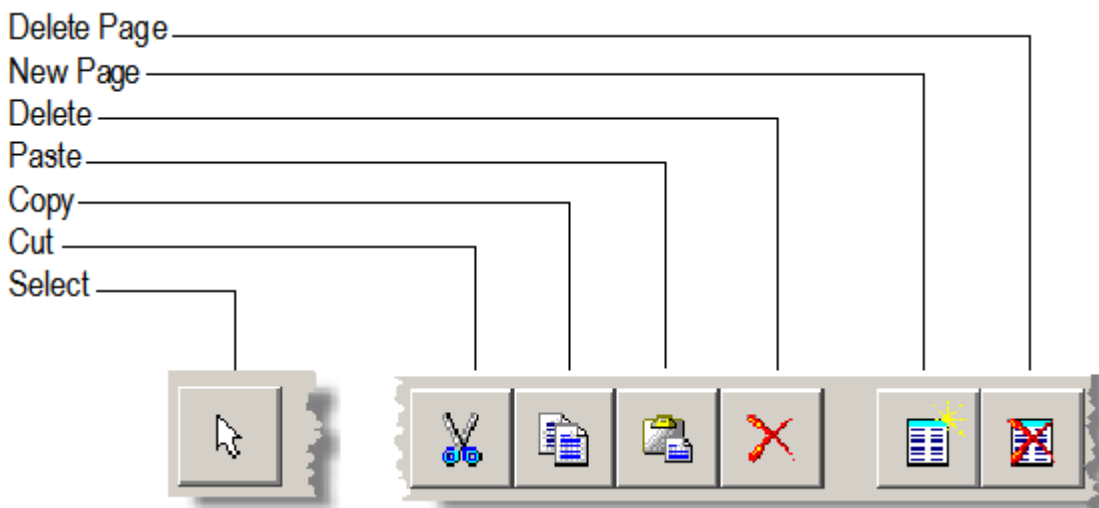
Page Menu



The Page Menu has the tools that create and delete pages in a TUI, as well as the tools to change the order in which they show up as tabs along the top of the TUI Design window in multiple page TUIs.



Tools and Editing



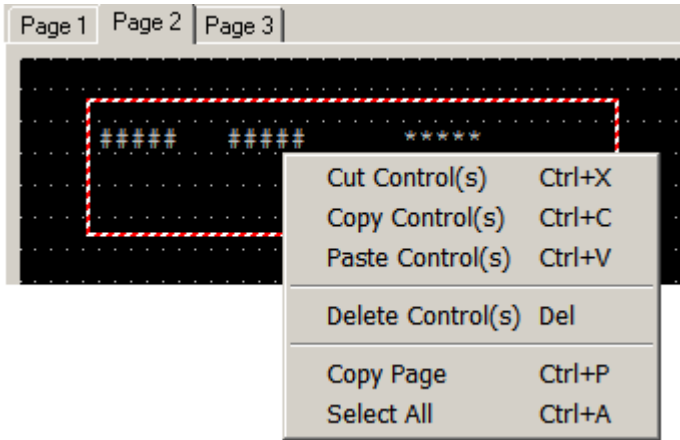
When you select a Control in the Design Window and click on the “Cut” button, the Control is erased, but saved on the “clipboard”. You can also simply copy a Control to the clipboard by selecting it and clicking on the “Copy” button. A Control on the clipboard can be placed in the Design Window with the “Paste” button . To delete a Control without putting it on the clipboard, select the Control and click on the “Delete” button.

There are a few handy editing functions that are not obvious to the new user of the TUI editor.

Rubber Band

Clicking and moving the mouse in the TUI designer will cause a selection "rubber band" that allows for the selection of multiple controls. Once selected these controls may be moved on the page, copied as a group to another page or deleted as a group.

The Right Click



Once a control(s) are selected, right clicking on the selected controls will bring up a speed menu with more tools.

You can cut, copy, paste or delete the controls selected or select and copy the page. Pages may be copied to one another and between Nodes in the Scadabuilder project.

General Settings

The settings found on the General tab allow you to specify parameters that apply to the TUI in general, as opposed to a specific page or control. These parameters can be overridden by parameters for a specific page or control.

General Page Control	
General	
Name	Tui 1
Port	(console on com1)
Port (2nd)	(none)
Mode	terminal
Header	My Header
Columns	80
Rows	25
Refresh (sec)	30
Scan rate (ms)	100
Auto edit	no
Quit enable	no
Security reg	[none]
Telnet port	23
Telnet max	1
Telnet timeout (min)	30
Scheme	color
Head attr	(Attributes)
Text attr	(Attributes)
Link attr	(Attributes)
Select attr	(Attributes)
Value attr	(Attributes)
Graph attr	(Attributes)
Button attr	(Attributes)
Disable attr	(Attributes)

Name

Used to identify the TUI, so it can be referenced elsewhere in your setup.

Port

Determines which port which the TUI should be displayed on. The port can be either a serial communications port or a Telnet port (TCP/IP port 23). The default is the console port, which is almost always Com1. Any other port selections must be first configured via a Network Port.

Port (2nd)

The TUI may be applied to a second port either serial or TCP/IP (telnet). This allows the use of the same TUI on multiple interfaces without having to recreate the same interface twice.

This can be very useful for simultaneously having a local and a remote TUI interface.

Mode

Determines the operational mode of the TUI.

- Terminal* Terminal mode is for connection to an ANSI-BBS compatible terminal or terminal emulator (such as HyperTerminal). If you will be displaying the TUI on your PC, set the mode to Terminal.
- Hotlink* Hotlink mode is for connection to ICL's HotLink operator interface terminal (an 8 by 21 character LCD display with integrated keypad).
- Viewpoint* Viewpoint mode is for connection to ICL's ScadaFlex ViewPoint operator interface terminal (a 4 by 20 character display with integrated keypad).



NOTE: You should also adjust the Columns and Rows parameters to match the size of your display device.

- Choose 80 columns/25 rows for a typical PC terminal emulator.
- Choose 21 columns/8 rows for a HotLink display.
- Choose 20 columns/4 rows for a Viewpoint display.

Header

An optional header can be displayed on the top line of each TUI page. The header is the same for all pages. The Header parameter sets the text that is to be displayed.

Columns

The Columns parameter should be set to match the number of columns that your display device can accommodate. Typical values would be 80 for a PC terminal emulator, 21 for the HotLink and 20 for the Scadaflex Viewpoint. The grid shown on the right side of the TUI Designer window reflects the current Columns setting.

Rows

The Rows parameter should be set to match the number of rows that your display device can accommodate. Typical values would be 25 for a PC terminal emulator 8 for the HotLink and 4 for the Scadaflex Viewpoint. The grid shown on the right side of the TUI Designer window reflects the current Rows setting.

Refresh

Determines how often (given in seconds) the TUI will periodically refresh the display. If the Refresh parameter is set to zero, it has the effect of disabling the refresh. The operator can cause a manual refresh by pressing the Delete key (in terminal mode) or the CLR key (in HotLink mode).

Scan Rate

On an operating TUI, all the currently displayed registers are periodically examined for changes that need to be reflected on the display. This setting determines how often (in milliseconds) the registers are examined. A setting of 0 is interpreted to mean "as fast as possible."

Auto Edit

Applies only to terminal mode. If enabled, does not require the operator to press Enter first before editing a Register Field or Buffer Field value. This setting may be overridden in an individual Register Field or Buffer Field control.

Quit Enable

Determines if the Ctrl+Q key, when pressed, will terminate the Virtual Machine and return to the operating system. This is an advanced feature that normally may be left disabled.

Telnet Max

This parameter is used whenever the TUI is applied to a TCP/IP port. It is particularly useful over Ethernet. It allows multiple users (up to 4) to be connected to a TUI at the same time.

Telnet Port

This parameter allows the Telnet interface to move from the TCP/IP port of 23 to another port for security purposes. Be careful not to use ports that other interfaces use such as the following defaults:

<i>FTP (File Transfer Protocol)</i>	21 and 20
<i>ISaGRAF</i>	1100
<i>Modbus</i>	TCP/IP 502
<i>HTTP</i>	80

Security Register

The Security Register allows the programmatic control of access to difference levels of the TUI interface. Each Group, Page and Control (these are TUI elements) of the TUI has a hard coded security level parameter. If the Security Register has a value equal to or greater than that of the element in question, that element is then enabled to the current interface.

Telnet Timeout

Allows the default time of 30 minutes to be overridden with another value. A zero in this field never times out.

Scheme

Sets the color and attribute scheme used by the TUI -- "black and white" (for monochrome terminals and the HotLink) or "classic" (for color terminals).

Colors and attributes may be overridden on the General settings tab and on individual pages and controls.

Security Level

Sets the security level as referenced by the Security Register. If the Security Register value is less than this level then the TUI Control will not be accessible.


See the TUI - Security Register section (TUI - General Tab) ***Security Register*** (on page 395) section for more information.

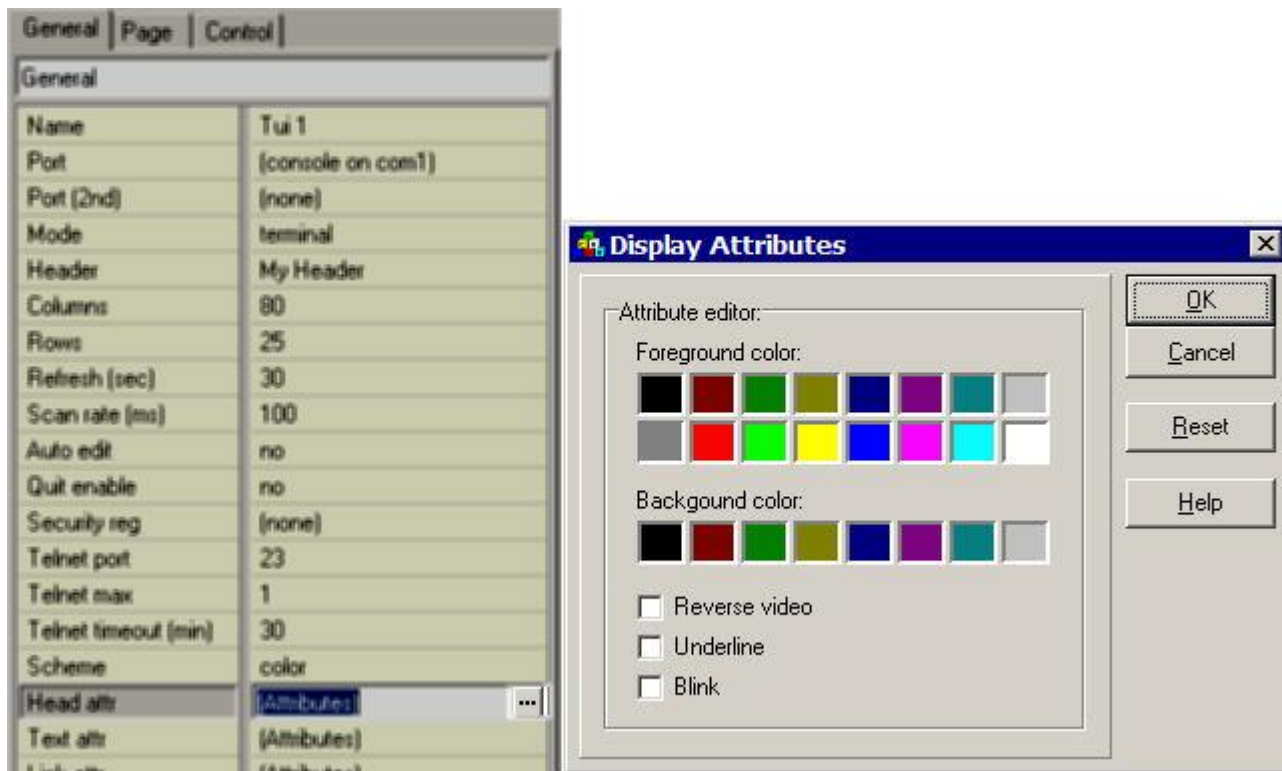
Attributes

Attributes are the look and feel of each controller in the interface and all TUI Controls have them. They can be set generally for the whole Interface, set by the page, or individually set for each control.

There are three main components for attributes:

- Foreground Color
- Background Color
- Disposition (reverse video, underline and blink)

Each of these is configured in the attribute dialog. To get to the attribute dialog, click on the ellipsis () on any field in the parameters section of the TUI designer.



Header Attributes

Sets the attributes used for display of the page header.

Text Attributes

Sets the attributes used for the display of static text. Can be overridden on individual pages and controls.

Link Attributes

Sets the attributes used for the display of Page Links.

Selection Attributes

Attributes used for display of the selector as you move around the page with the cursor keys.

Value Attributes

Attributes used for display of values. Can be overridden on a page or on individual controls.

Graph Attributes

Attributes used for display of Bar Graphs. Can be overridden on individual Bar Graphs.

Button Attributes

Attributes used for display of TUI Buttons. Can be overridden on a page or on individual Button controls.

Disable Attributes

Attributes used for display of Page Links when the link is disabled (via the disable register).

Page Settings

The settings found on the Page tab allow you to specify parameters that apply in general to the currently selected page. These parameters can be overridden by parameters for a specific control.

Page	
Name	Page 2
Prompt	
Group	Main Level Group
Security level	0
Text attr	(Attributes)
Link attr	(Attributes)
Value attr	(Attributes)
Button attr	(Attributes) ...

Name

Used to identify the TUI page, so it can be referenced by a Page Link control on another page.

Prompt

When this page is referenced by a Page Link on another page, the prompt is displayed for the link text.

Group

Page Groups allow the configuration of Group levels within the TUI interface. For example, if a series of pages belong to groups like the following:

Group1

Page1

Page2

Page3

Group2

Page4

Page5

Page6

Once a page has been navigated to, the pages in the group--Pages 1, 2, and 3--may be accessed with the Prev and Next buttons (ScadaFlex ViewPoint) or the Ctrl P, Ctrl N (terminal interface) keys. A page from one group must have a link to another page in another group for the second group to be accessed from the interface.

A page may be moved from or to any group within the interface.

Security Level

Sets the security level as referenced by the Security Register. If the Security Register value is less than this level then the page will not be accessible.


See the TUI - **Security Register** (on page 395) section (TUI - General Tab) for more information.

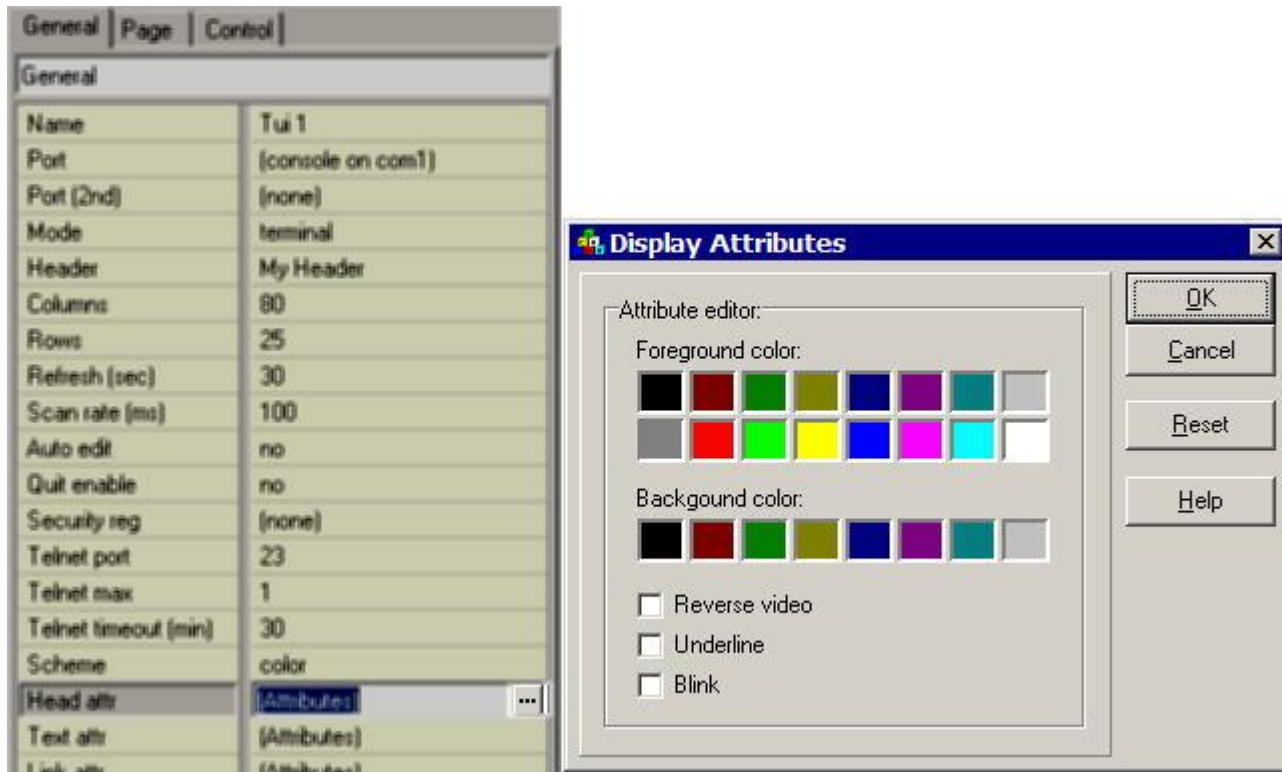
Attributes

Attributes are the look and feel of each controller in the interface and all TUI Controls have them. They can be set generally for the whole Interface, set by the page, or individually set for each control.

There are three main components for attributes:

- Foreground Color
- Background Color
- Disposition (reverse video, underline and blink)

Each of these is configured in the attribute dialog. To get to the attribute dialog, click on the ellipsis () on any field in the parameters section of the TUI designer.



Text Attributes

Attributes used for display of static text. Overrides any settings on the General tab.

Value Attributes

Attributes used for display of values. Overrides any settings on the General tab.

Button Attributes

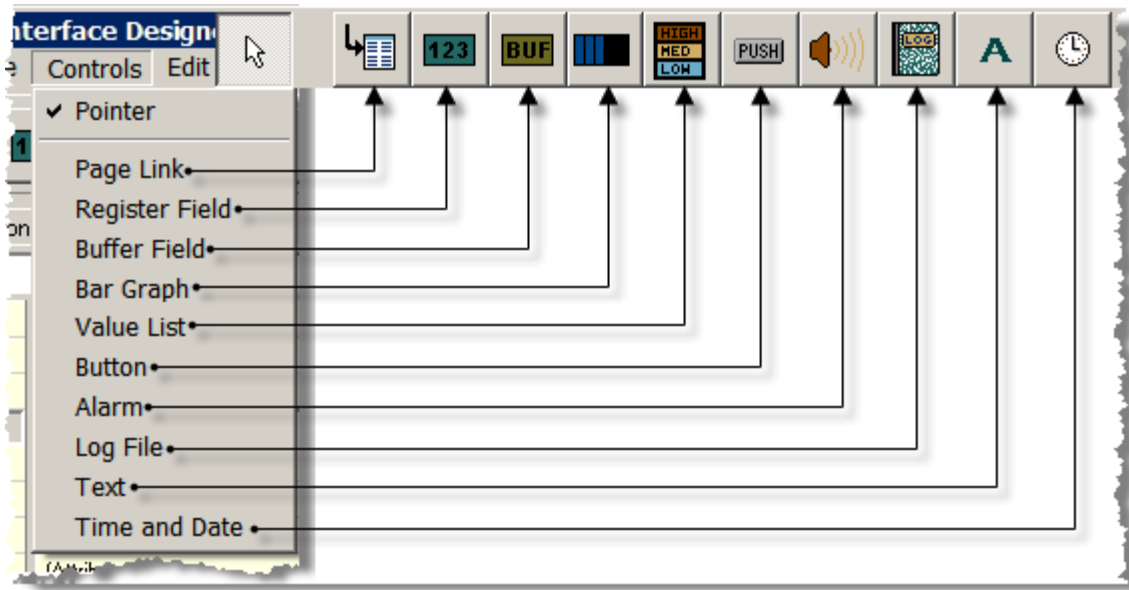
Attributes used for display of TUI Buttons. Overrides any settings on the General tab.

Page Link Attributes

Attributes used for display of TUI Page Links. Overrides any settings on the General tab.

Controls

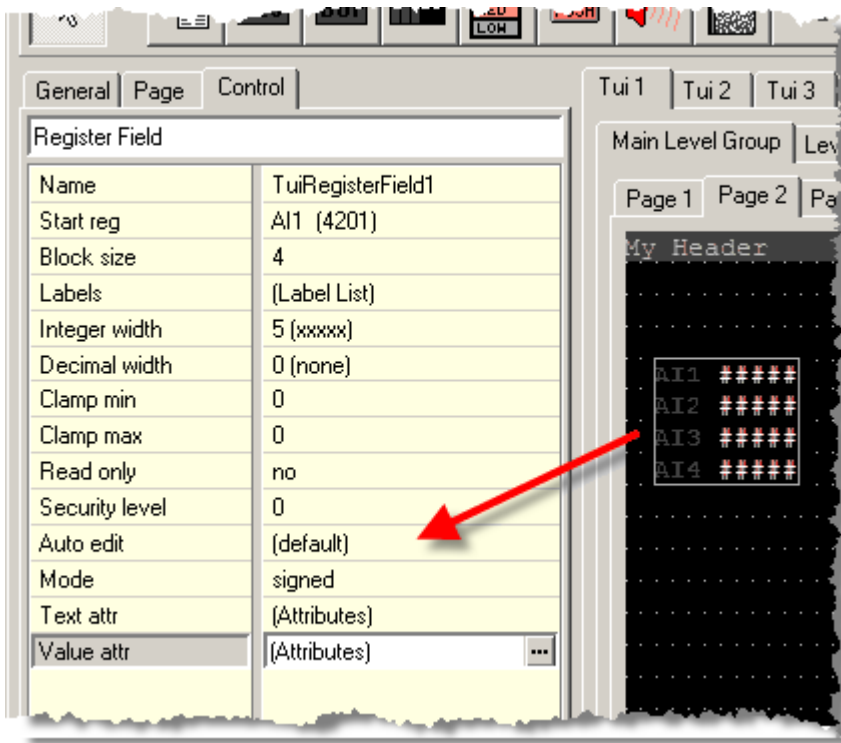
The TUI Controls are the design elements on a TUI page. They are grouped together under the “Controls” menu. There are also “Hot Buttons” for each element which are usually faster and easier to get to when creating a TUI page. Here are the menu and button insert selection for each control.



To insert a control, select it from the Controls menu or click on the control's hot button. Click in the TUI designer where you want the upper left hand corner of the control to be.

Controls can be moved at any time by selecting them and dragging them.

Control Parameters



For each TUI control, there is a set of configuration parameters that will be displayed under the "Control" tab to the left side of the TUI Design Window after the Control has been placed and selected in the Design Window. A Control is automatically selected just after it is created, or when you click on it with the "Pointer" tool selected. The selected Control is highlighted with a cyan color border.

TUI - Alarm Settings

A TUI Alarm control is used to represent the state of an Alarm.

Name

Used to identify the TUI control, so it can be referenced elsewhere in your setup.

Alarm

Identifies the Alarm for which the state is to be display.

Label

The Label text is placed in front of the alarm state on the TUI display.

Unack Text

Defines the text that is to be displayed when the Alarm in the unacknowledged state (the Alarm has been triggered but not yet acknowledged).

Ack Text

Defines the text that is to be displayed when the Alarm in the acknowledged state (the Alarm has been acknowledged but the alarm condition still exists).

Idle Text

Defines the text that is to be displayed when the Alarm in the idle state (the alarm condition is not active and any previous alarm condition has been acknowledged).

Unack Attr

Attributes used for the Alarm display when the Alarm is in the unacknowledged state.

Ack Attr

Attributes used for the Alarm display when the Alarm is in the acknowledged state.

Idle Attr

Attributes used for the Alarm display when the Alarm is in the idle state.

TUI - Bar Graph Settings

A Bar Graph control is used to represent register values as horizontal sliding indicators. Multiple registers may be shown, each with its own bar, by using a Block Size greater than 1.

When 'Terminal' or 'HotLink' mode is used for the TUI (under general settings), the graph control will use solid sliding bands to represent the register values. In 'Viewpoint' mode a series of asterisk characters will be used to represent the register values.

Name

Used to identify the Bar Graph control, so it can be referenced elsewhere in your setup.

Start Register

The starting register that defines the source of data to display. If the Block Size is set to greater than 1, additional sequentially indexed registers will be displayed, with one bar per register.

Block Size

The number of registers to display. You should not specify a number greater than what would be possible to show on the display.

Labels

A list of text labels to be displayed to the left of the bars. One label is displayed for each bar.

Width

The number of columns to use for the bar to represent its full-scale value.

Range

The Range Min and Range Max determine the range of values that the Bar Graph represents. When the value is at or below the minimum, no asterisks will be displayed. When the value is at or above the maximum, the bar will be filled with asterisks.

Text Attributes

Attributes used for display of text labels. Overrides General and Page settings.

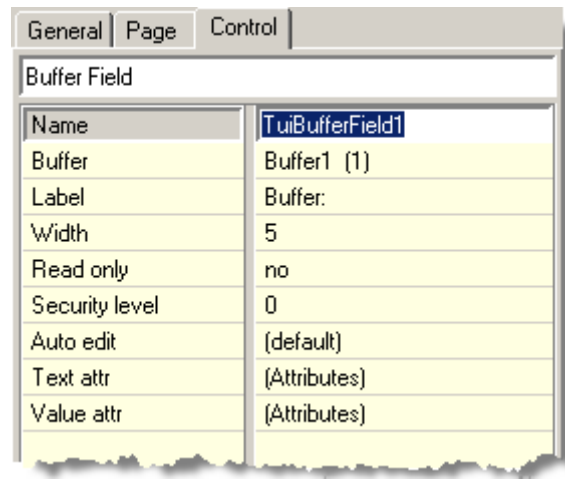
Graph Attributes

Attributes used for display of the bar indicator itself. Overrides General and Page settings.

When 'Terminal' or 'HotLink' mode is used for the TUI (under general settings), the graph control will use solid sliding bands to represent the register values. In 'Viewpoint' mode a series of asterisk characters will be used to represent the register values.

TUI - Buffer Field Settings

The Buffer Field control defines an area for displaying a buffer type register. If desired, the Buffer Field can be writable so the operator can input or change values.



Name

Used to identify the Buffer Field, so it can be referenced elsewhere in your setup.

Buffer

The buffer register to be displayed.

Label

A text label that is to be displayed to the left of the buffer value.

Width

The number of characters of the buffer value that should be displayed.

Read Only

Determines if the Buffer Field should be read-only, or read/write. If Read Only is set to "no," the operator can change the value.

Auto Edit

Applies only to terminal mode. If enabled, does not require the operator to press Enter first before editing the value. This may be used to override the setting on the General TUI settings tab.

IP Mode

Setting this option to Yes allows buffers to be edited like IP addresses. Only numeric entries are allowed. A "." or tab during entry will move the user to the next IP field. Enter will stuff the edited values in the Message (buffer) register.

Value Attributes

Attributes used for display of buffer values. Overrides General and Page settings.

Text Attributes

Attributes used for display of buffer field labels. Overrides General and Page settings.

Security Level

Sets the security level as referenced by the Security Register. If the Security Register value is less than this level then the TUI Control will not be accessible.

See the TUI - Security Register section (TUI - General Tab) *Security Register* (on page 395) section for more information.

TUI - Button Settings

A TUI Button defines an area on the screen that the operator can select and activate. The button name can be used wherever Triggers can be used. Essentially, a TUI Button is a Trigger.

Button	
Trigger name	Alarm Ack Button
Prompt	<Alarm Ack>
Security level	0
Key code reg	(none)
Key prompt	
Key timeout	200
Button attr	(Attributes)

Name

Used to identify the TUI Button, so it can be referenced elsewhere in your setup.

Prompt

The prompt is displayed to represent the Button on a TUI page.

Security Level

Sets the security level as referenced by the Security Register. If the Security Register value is less than this level then the TUI Control will not be accessible.

See the TUI - Security Register section (TUI - General Tab) *Security Register* (on page 395) section for more information.

Key Code Register

When this specialized parameter is used, the key code of each key press from the attached terminal keyboard is stored to the specified register when the user has the TUI Button selected.

The key code is cleared to zero after the time entered into the Key Timeout parameter has expired.

Key Prompt

When a Key Code Register has been selected, the Key Prompt will be displayed on the TUI Button when the operator presses Enter on the Button. The prompt stays until the operator presses Enter again.

Key Timeout

Used when a Key Code Register is specified for a TUI Button. When a key is pressed, the register will be loaded with the corresponding key code. After the Key Timeout, the register will be zeroed.

Button Attributes

Attributes used for display of TUI Buttons. Overrides General and Page settings.

TUI - Log Settings

A TUI Log control displays an alarm or event log. The display area shows the most recent entries added to the log, and dynamically updates as entries are added. Pressing the Enter key while in the log area will place the display into scroll mode allowing you to view previous entries that have been buffered.

Log	
Name	TuiLog2
Log file	MyAlarmLog
Width	40
Height	5
Border	box
Title	Alarm Log
History	0
Security level	0

Name

The name associated with the TUI Log control. This parameter is not required but can be a helpful identifier.

Log File

Allows you to select the Log File to display.

Width

The number of columns to be used for the log display.

Height

The number of rows to be used for the log display.

Security Level

Sets the security level as referenced by the Security Register. If the Security Register value is less than this level then the TUI Control will not be accessible.

See the TUI - Security Register section (TUI - General Tab) **Security Register** (on page 395) section for more information.

Border

The type of border to display in the log area:

- None - no border is displayed
- Box - a box is displayed around the log data
- Top - a line is displayed above the log data
- Bottom - a line is displayed below the log data

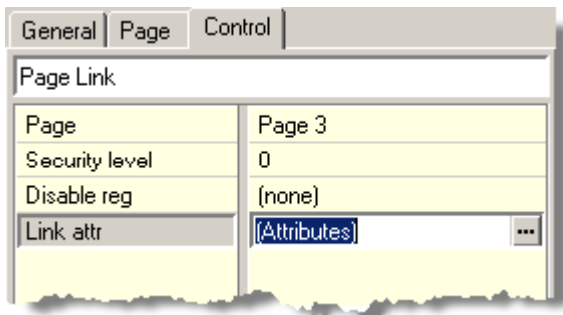
Title

The optional Title is displayed as a label for the log data.

History

Specifies the number of past log entries (since the application was started) that will be buffered for viewing in the display area.

TUI - Page Link Settings



A Page Link takes the operator to another TUI page when he moves the cursor to the Page Link and presses Enter. This can be used to create menu systems and "hypertext links" similar to a web page.

Page

Sets the page that is linked to. The Prompt parameter on the linked page will be displayed, if defined, otherwise the page name is used.

Disable Register

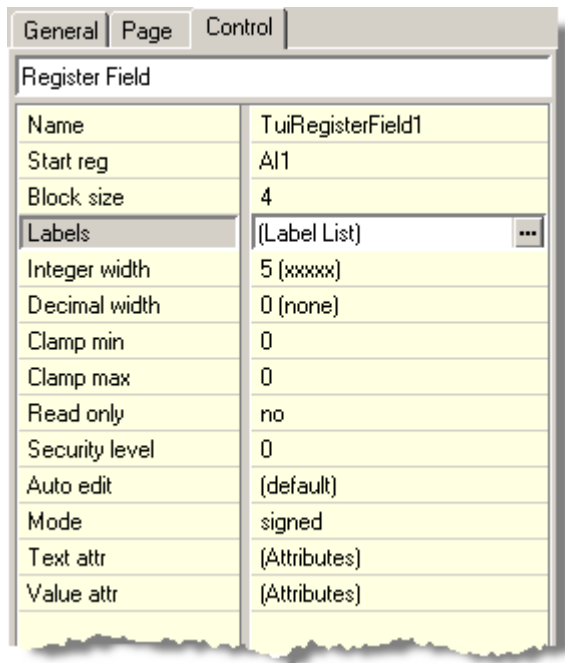
A boolean register can be used to disable access to a Page Link. This can be used for security purposes, for instance. If the register has a 1 (true) value, the Page Link will be disabled.

Security Level

Sets the security level as referenced by the Security Register. If the Security Register value is less than this level then the TUI Control will not be accessible.

See the TUI - Security Register section (TUI - General Tab) **Security Register** (on page 395) section for more information.

TUI - Register Field Settings



Register Field	
Name	TuiRegisterField1
Start reg	A11
Block size	4
Labels	(Label List) ...
Integer width	5 (xxxxx)
Decimal width	0 (none)
Clamp min	0
Clamp max	0
Read only	no
Security level	0
Auto edit	(default)
Mode	signed
Text attr	(Attributes)
Value attr	(Attributes)

The Register Field control defines an area for displaying a vertically oriented list of values from one or more registers. If desired, the Register Field can be writable so the operator can input or change values.

Name

Used to identify the Register Field, so it can be referenced elsewhere in your setup.

Start Register

The starting register that defines the source of data to display. If the Block Size is set to greater than 1, additional sequentially indexed registers will be displayed.

Block Size

The number of registers to display. You should not specify a number greater than what would be possible to show on the display.

Labels

A list of text labels to be displayed to the left of the list of registers in the Register Field. One label is displayed for each register.

Security Level

Sets the security level as referenced by the Security Register. If the Security Register value is less than this level then the TUI Control will not be accessible.

See the TUI - Security Register section (TUI - General Tab) **Security Register** (on page 395) section for more information.

Integer Width

The number of digits of integer information to display. Determines the number of digits shown to the left of the decimal point (if any) on floating point registers, or the total number of digits for an integer register.

Also determines the number of digits shown for a Boolean register. Boolean values are shown as "1" and "0" by a Register Field. Tip: if you wish to show text such as "ON" and "OFF" for a Boolean register, you can use a Value List.

Decimal Width

For floating point registers, determines the number of digits shown to the right of the decimal point.

Clamp

Determines the range of values that the operator is allowed to enter for this field.

Read Only

Determines if the Register Field should be read-only, or read/write. If Read Only is set to "no," the operator can change the value(s).

Auto Edit

Applies only to terminal mode. If enabled, does not require the operator to press Enter first before editing a value. This may be used to override the setting on the General TUI settings tab.

Mode

Sets the mode used to display and input values of integer type fields. Overrides the default mode that is initially set based on the register type.

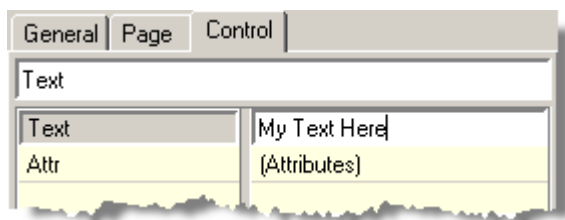
Text Attributes

Attributes used for display of text labels. Overrides General and Page settings.

Value Attributes

Attributes used for display of register values. Overrides General and Page settings.

TUI - Text Settings



Static Text can be placed anywhere on a TUI page. Static Text is for display purposes only -- the operator cannot select or interact with it.

TUI Text

The text that is to be displayed. Attributes for Text display can be selected on the Page or General tabs.

Text Attributes

Attributes used for display of text labels. Overrides General and Page settings.

TUI - Time and Date Settings

Time/Date	
Format	mm-dd-yyyy
Read only	yes
Security level	0

A time or date field can be placed on a TUI page. The field can be read-only or writeable. If it is writeable, this provides a means for the operator to set the time and date.

Time and Date - Format

The format that you would like to use for display of the time or date. If you want both time and date, put two separate fields on the page and select date for one and time for the other.

Format	Meaning
hh:mm:ss xx	hours, minutes, seconds and AM/PM
hh:mm:ss	hours, minutes and seconds
hh:mm xx	hours, minutes and AM/PM
hh:mm	hours and minutes
mm-dd-yyyy	month, day and 4-digit year
mm-dd-yy	month, day and 2-digit year

Time and Date - Read Only

If you would like the date/time field to be read-only, set this parameter to "yes." If you would like to allow the operator to set the date/time through the TUI, set Read Only to "no."

Security Level

Sets the security level as referenced by the Security Register. If the Security Register value is less than this level then the TUI Control will not be accessible.

See the TUI - Security Register section (TUI - General Tab) **Security Register** (on page 395) section for more information.

TUI - Value List Settings

A Value List control defines a list of text strings that are associated with register values. When the value in the associated register changes, the displayed text will automatically change correspondingly. Value Lists can be made selectable – the operator can select the different text strings using the Up/Down arrow keys. When the Enter key is pressed the value associated with that text string will be written to the register.

Value List	
Name	TuValueList1
Start reg	A11 (4201)
Block size	6
Labels	[Label List] ...
Values	[Value List]
Read only	no
Security level	0
Auto select	no
Text attr	[Attributes]

Name

Used to identify the Value List control, so it can be referenced elsewhere in your setup.

Start Register

The starting register that defines the source of data to display. If the Block Size is set to greater than 1, additional sequentially indexed registers will be displayed, with one row per register.

Block Size

The number of registers to display, one row per register. You should not specify a number greater than what would be possible to show on the display.

Labels

A list of text labels to be displayed to the left of the value text. One label is displayed for each row.

Values

The value list editor allows you to set up a list of values and corresponding text to be displayed on the TUI. To set up a list:

- Enter a label into the Text box. This first label represents values that are less than the lowest threshold. Click Add.
- Enter another text label and the corresponding threshold value. If the register value is equal to or greater than this threshold value, the corresponding label will be displayed when the TUI is running. Click Add.
- Keep adding threshold values and corresponding text until you are done.

Security Level

Sets the security level as referenced by the Security Register. If the Security Register value is less than this level then the TUI Control will not be accessible.

See the TUI - Security Register section (TUI - General Tab) **Security Register** (on page 395) section for more information.

Read Only

Determines if the Value List should be read-only, or read/write. If Read Only is set to "no," the operator can change the value(s) using the Up/Down arrow keys when the Value List is selected on the TUI.

Auto Select

Affects how the operator interacts with a Value List is read-only is turned off.

Normally, to modify a value list, the operator moves to that list and presses the Enter key to access edit mode. Once in edit mode, the following keys apply:

Keys	Description
Down arrow	Select previous entry from list.
Up arrow	Select next entry from list.
Enter	Accept new entry and load value in register.
Esc	Cancels edit and restore last entry.

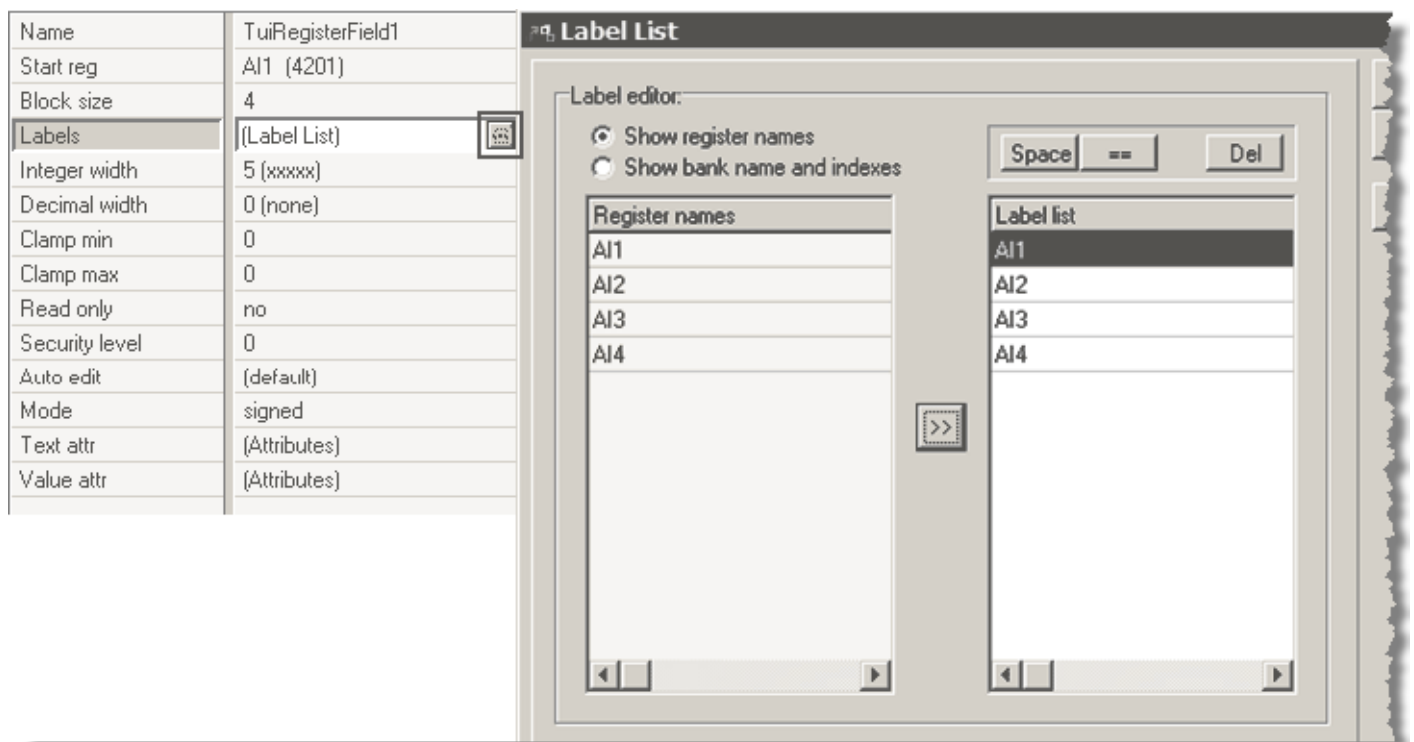
If Auto Select mode is enabled for a Value List, the values can be stepped through by pressing Ctrl+S (in terminal mode) or the Star (*) key (on the HotLink). With each key press the next entry from the list will be selected, loading the associated value into the register.

Text Attributes

Attributes used for display of text labels. Overrides General and Page settings.

TUI Designer - Label List Editor

The Label List Editor allows you to create and edit a list of labels that will be displayed on the TUI.



The right half of the window shows the actual labels that will be displayed. The left half of the window displays a list that can be optionally copied to the right half by pressing the ">>" button. This provides a quick way to create a label list if you want to use the register names or bank names and indexes.

To have something displayed on the left side, so it can be copied to the right, choose "Show register names" or "Show bank name and indexes."

To create or edit labels shown on the right side, simply click on the row you want and start typing.

Show Register Names

Shows register names on the left side of the window. These are then available for copying to the right side of the window with the ">>" button. The labels on the right side of the window are what will actually be displayed alongside the register values.

Show Bank Name and Indexes

Shows the register bank name and register indexes on the left side of the window. These are then available for copying to the right side of the window with the ">>" button. The labels on the right side of the window are what will actually be displayed alongside the register values.

Register List

This is a list of registers that you can copy to the right-hand side of the window, using the ">>" button. The right side of the window shows the actual labels that will appear next to the list of register values when the application runs. This provides a quick way to define the labels.

Labels

This is the list of labels that will be displayed next to the register values on the TUI when the application runs. You can edit a label by clicking on it and typing in the new text.

When the application is run, the width of the label column is determined by the widest entry. If you want to have some additional space between the labels and the values, you can put trailing spaces on one of the labels.

Edit Buttons

The edit buttons allow the label strings to be manipulated as a group. Alignment is always maintained at the end of the label strings. The edit buttons perform the following functions:

- The "Space" button inserts one space character at the end of each label.
- The "==" button inserts one equals character at the end of each label.
- The "Del" button removes one character from the end of each label.

>> Button

This button copies the registers shown on the left side of the window to the right side of the window. The right side of the window shows the actual labels that will be used.

TUI Designer - Value List Editor

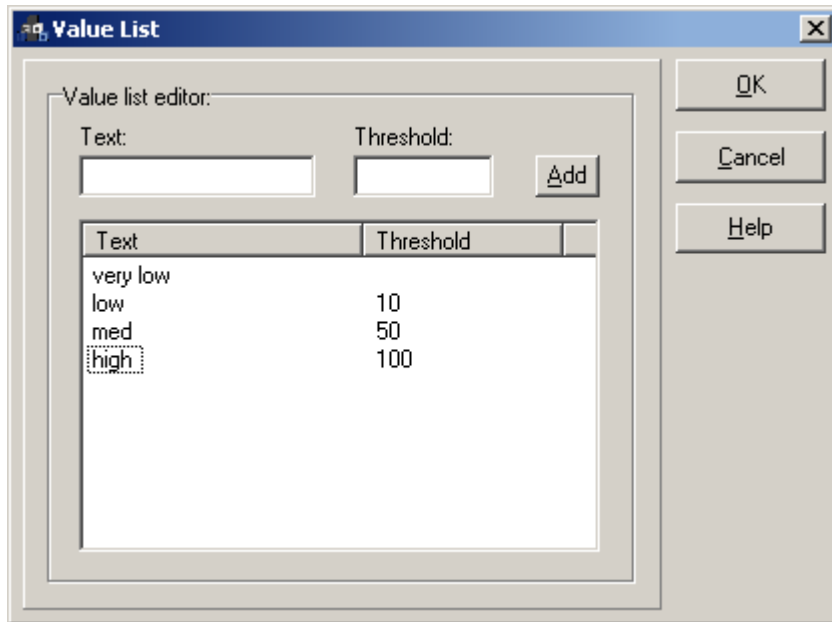
The Value List Editor allows you to create and edit a list of text strings that will be displayed on the TUI when the corresponding register value changes.

A Value List control defines a list of text strings that are associated with register values. When the value in the associated register changes, the displayed text will automatically change correspondingly.

Start by entering a text string and clicking the "Add" button. This string will be displayed as a "default." That is, when the register value is less than all of the threshold values that the list consists of, the default text will be displayed.

Next, enter another text string and corresponding threshold value. When the value is equal to or greater than the threshold, the text will be displayed. Enter more text/threshold pairs as desired.

For example, if you were to set up the following list:



The value list will cause the following text to be displayed under these conditions:

Condition	Display
value < 10	very low
value >= 10 and value < 50	low
value >= 50 and value < 100	med
value >= 100	high

You can change the colors/attributes of each text string by right-clicking on the text after it gets added to the list.

To delete or edit the text, right click on it as well.

Text

This is the text that will be displayed corresponding to the threshold value.

Threshold

The threshold value which causes the corresponding text to be displayed. To cause the corresponding text to be displayed, the register value must be equal to or greater than the threshold and less than the next higher threshold.

Add Button

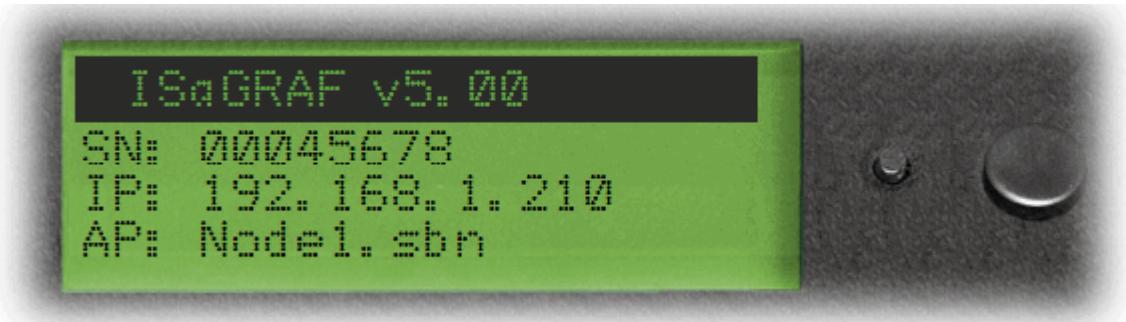
The "Add" button adds a text/threshold pair to the list.

List

This is the list of text strings and their corresponding threshold values. You can change the colors/attributes of each text string by right-clicking on the text. To delete or edit the text, right click on it and choose "Delete."

TUI Local HMI (Pinnacle Controllers)

The Local HMI of a Pinnacle Controller has several uses. In its default state it can show the current ISaGRAF Kernel version, Serial Number, IP Address, and current running application.



The HMI can also show status information and direct the user's efforts when changing USB flash drives or recovering and updating the unit.



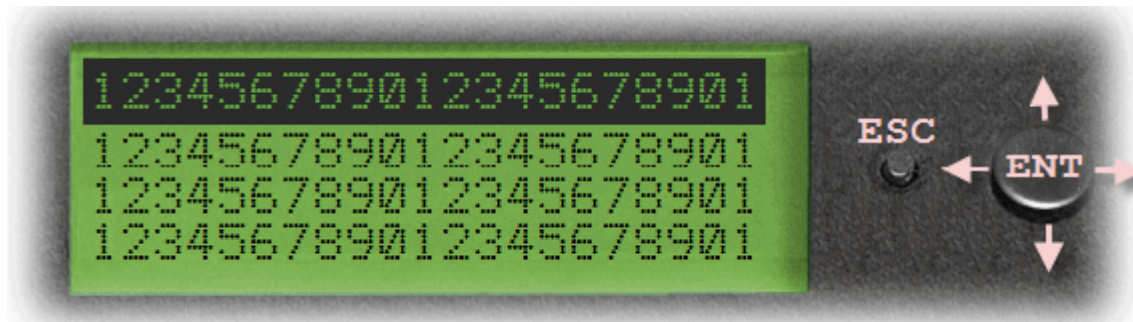
The HMI can offer much more to the user in the form of a complete user interface. The system is configured through the Textual User Interface (TUI) configuration tools just like a serial or Ethernet TUI. The same interfaces, display, and navigation principles apply.

In This Section

Local HMI Navigation.....	415
Creating A Local HMI.....	416
Local HMI Numeric Entry.....	421

Local HMI Navigation

Navigation takes place at several levels Screen, Page, Group and Global Levels. The keys for navigation are shown below. The button on the left is the Escape/Cancel key and the button on the right is actually a joy stick with a center down actuation that allows for Enter and Confirmation.



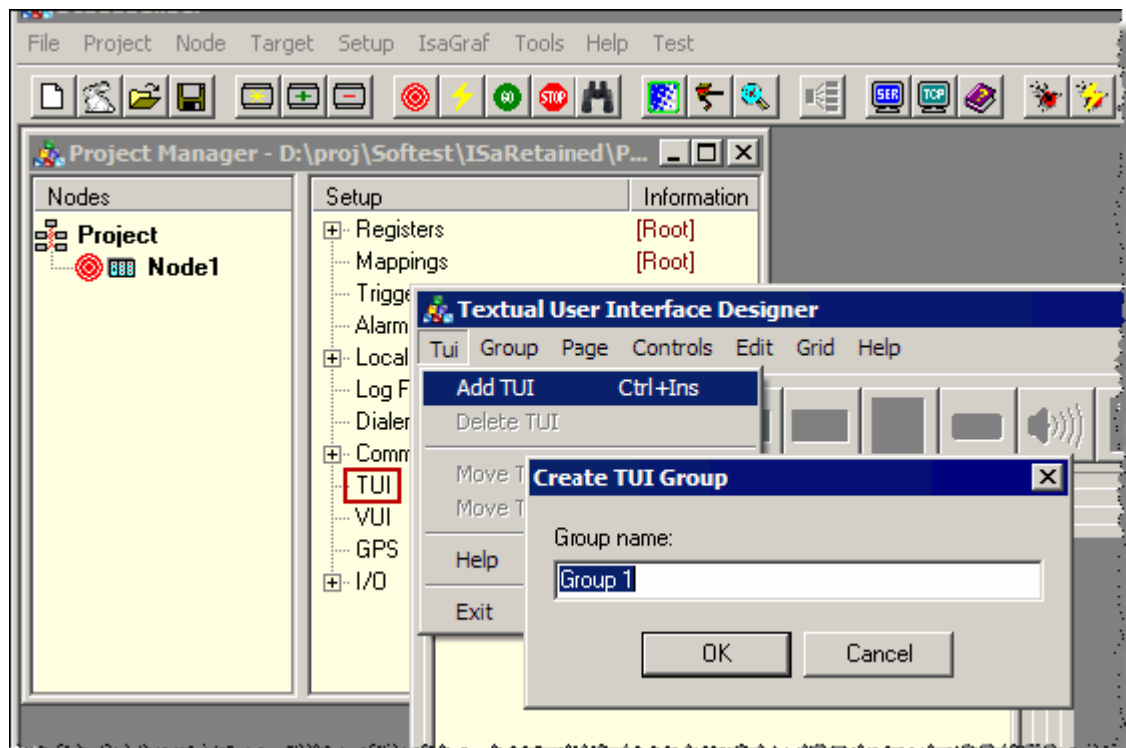
Pressing and holding the right or left arrow keys on the joy stick button will make the interface move to the next page in that direction.

Let's take a look at working with a screen first and create our HMI (TUI) from the ground up.

Creating A Local HMI

First we must create a TUI to place control on to:

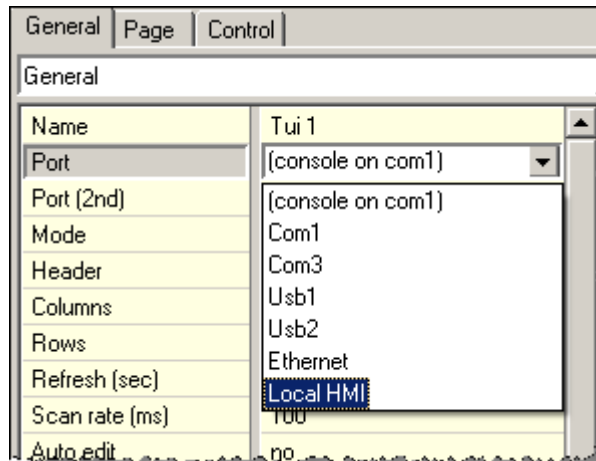
Double Click on the TUI item in the Setup Pane of ScadaBuilder



Click OK on Group1. We will use that later.

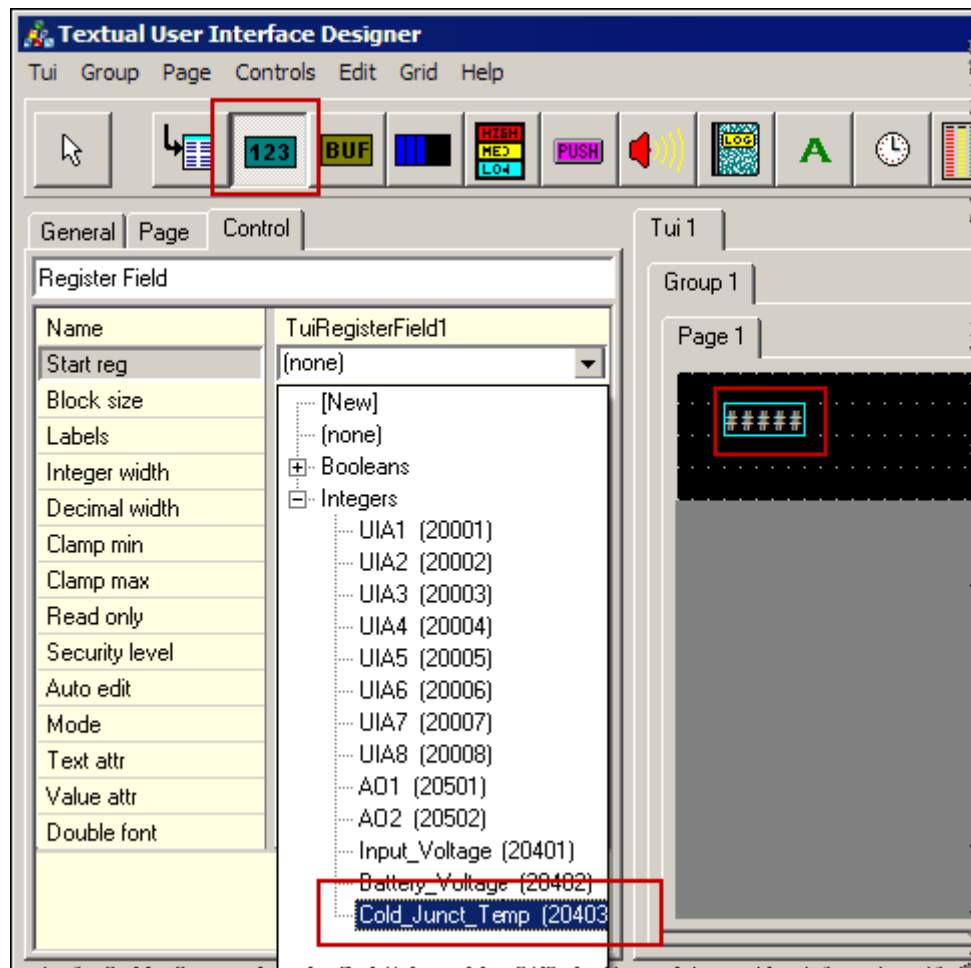
On the General Tab of the TUI, select the Local HMI option.

This will configure the TUI to the right size and interface options.



Click on the Register Button and click in the black interface area of the new TUI work space.

Select the Cold_Junct_Temp register to apply to this field.



Register Field	
Name	TuiRegisterField1
Start reg	(none)
Block size	1
Labels	(Label List)
Integer width	5 (xxxxx)
Decimal width	0 (none)
Clamp min	0
Clamp max	0
Read only	yes
Security level	0

Since this is an input only register and may not be modified, we need to set the Read Only attribute shown here.

When navigating on a screen, setting a field to read only will allow the navigation keys to skip to the next read/write field shown.

Download this configuration to the controller and you should see something like the following displayed on the controller:



Lets add a Label to this display and a Read/Write variable to the interface...

Click on the "#####" in the TUI display area...

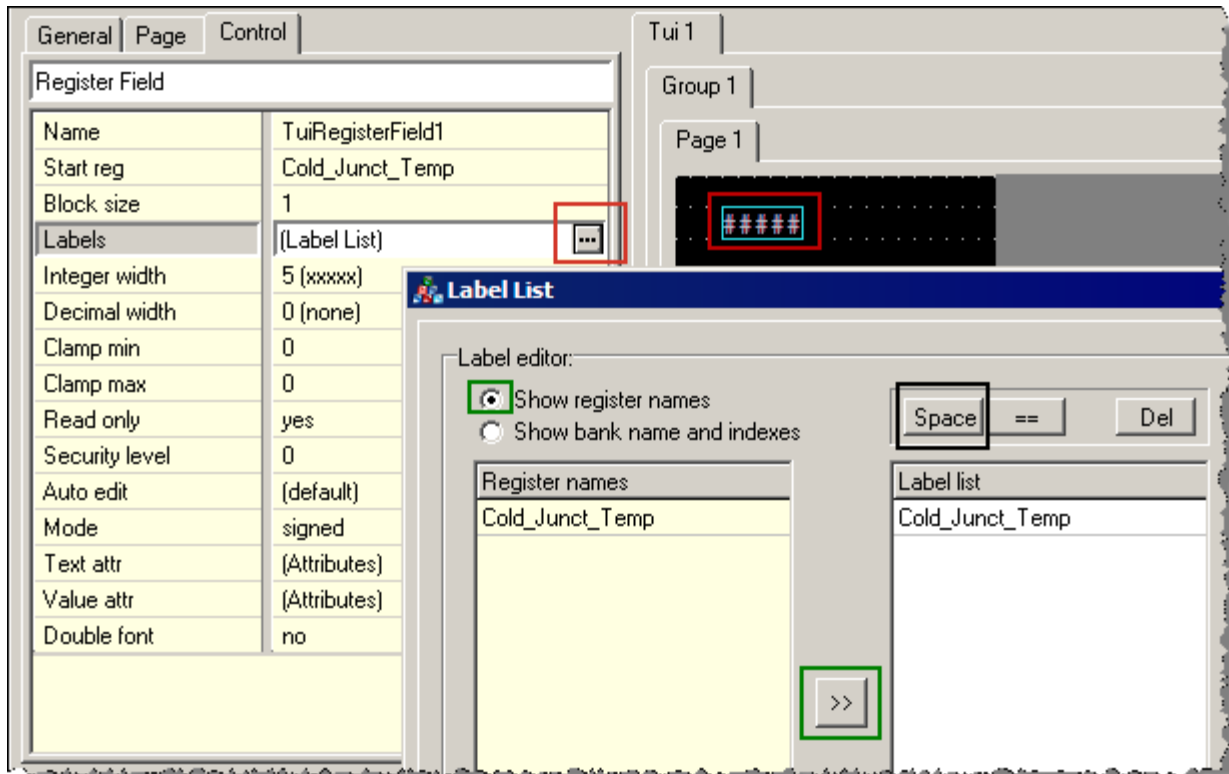
Click on "Label List ..."

Click the "Show register names..."

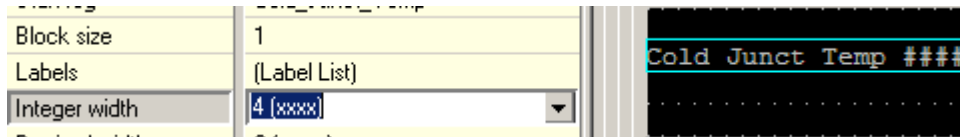
Click the ">>" box...

Click the "Space" button...

Click OK.



Move the display all the way to the right and set the "Integer Width" to 4(xxxx) so the display will fit on the small HMI screen:



This can be done with any register or block of registers in the system. Your TUI configuration should look like this above.

Your HMI screen should look something like this:



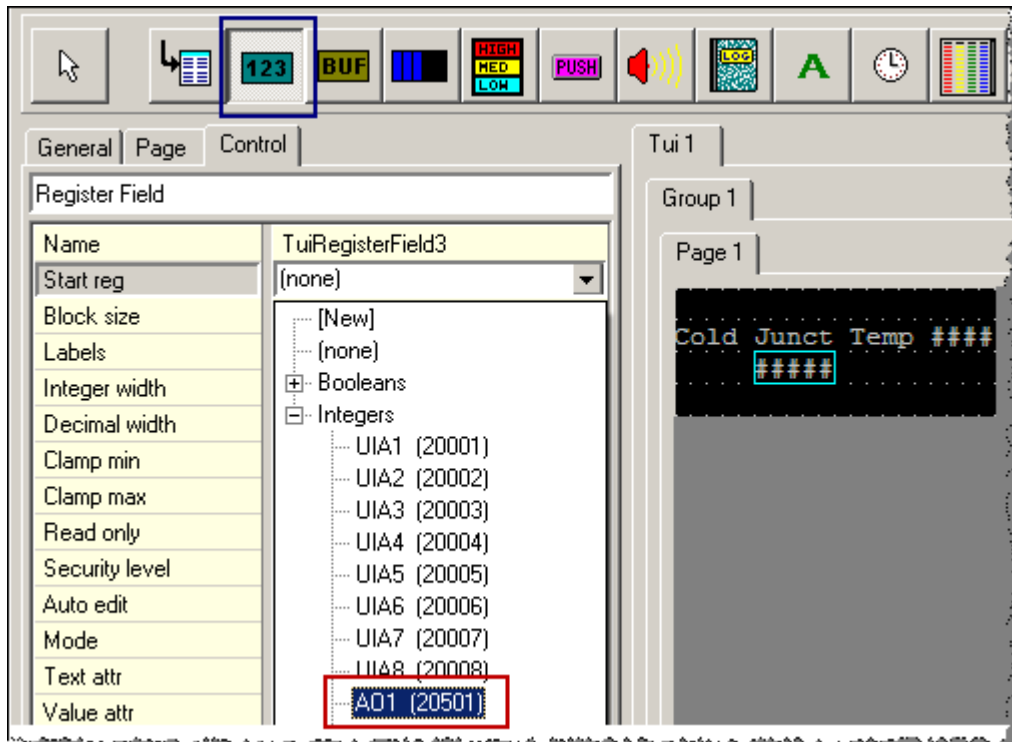
Let's do one more configuration before we move on to the next section so we have something to edit:

Insert another register, AO1 this time:

Click the Register Button...

Place the Register in the black TUI work space (leaving space for the "AO1_" label...

Select AO1 from the Integers register list...



Be sure the Read Only configuration is set to No...

Just like we did above:

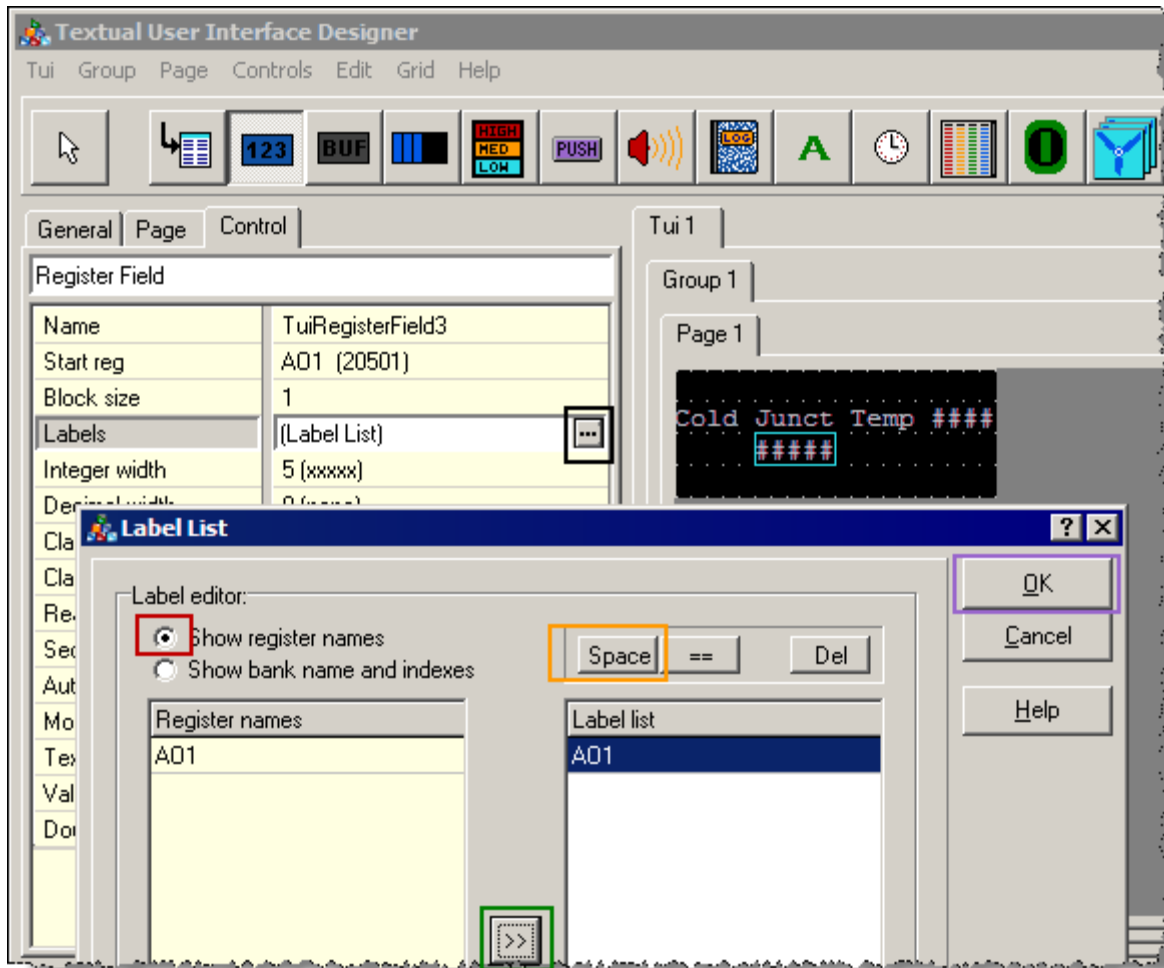
Click on "Label List ..."

Click the "Show register names..."

Click the ">>" box...

Click the "Space" button...

Click OK.



You should end up with a display configuration like this with AO1 as a Read / Write field:

```
Cold Junct Temp ####
AO1 #####
```

Download this to your controller. You should see a display like this:



Local HMI Numeric Entry

Now that we have a display we can use the joy stick to enter a numeric value.

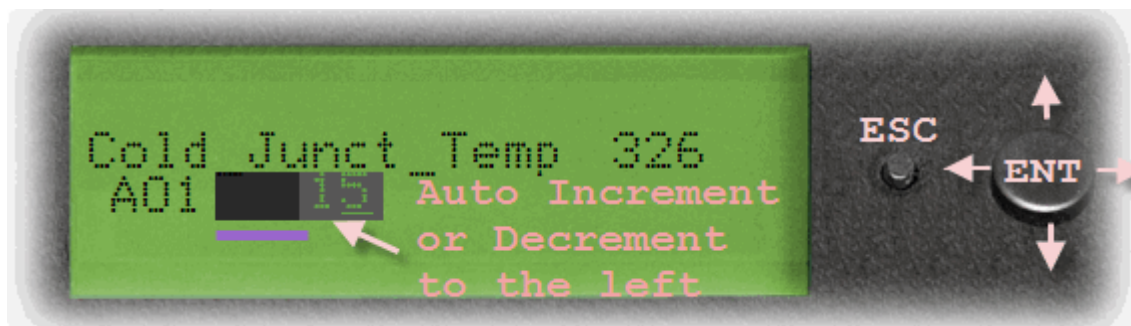


The navigation keys are shown below. To select a field for entry, use the arrow keys to navigate the Read / Write fields on the screen and highlight the desired field. Press the very center of the joy stick button straight down until you get a flashing cursor:



The up direction on the joystick will increment the place flashing by one until 9 is reached and will roll over that digit and then increment the next place to the left. Pressing and holding the up arrow will do this quickly after 2 seconds.

The down direction will decrement the the number in the same way rolling back to 9 when digit gets to 0.



You may also move to the left manually and modify any position in the field in the 10's, 100's, 1000's place and so on. Pressing and holding the down and up arrows works the same way in any of these positions leaving the lower positions alone. This way, large or small changes are possible with a minimum of key presses.

To finish the edit, press the button straight down to actuate an enter (ENT). To abort the edit, press the ESC key.

User Portal (Web Interface)

Web pages are provided on the Pinnacle controllers to access a variety of display, editing and maintenance functions that can either be done locally or remotely from a TCP/IP connection through any web browser. Simply type in the IP address of the controller into the address bar of the web browser and you will be asked for a login. Once the login is complete, the user account controls access and allows functions to be available.

Once logged in, the User Portal will remember the current login and use it so long as the controller deems it valid. If the controller gets reset or the User Portal does not see activity for 10 minutes, the User Portal will post an error and go back to the User Login... dialog. If the page is refreshed from the browser, these login settings are remembered for 24 hours.

The User Portal is setup in a Link / Dialog / Wizard format. Access is not available unless the controller deems that the current user has permissions. The User Portal requires no browser add-ons such as Java or Adobe Flash. This is done to maintain compatibility for most Javascript compatible browsers. This include but are not limited to:

- Internet Explorer 7 and greater.
- Firefox
- Google Chrome
- Opera
- Safari

Before we can begin, an administration account will need to be setup in ScadaBuilder and the downloaded to the controller. From there, user accounts control what functions are available to users in the User Portal.

The User Portal web interface is only available on Pinnacle and later controllers.

In This Section

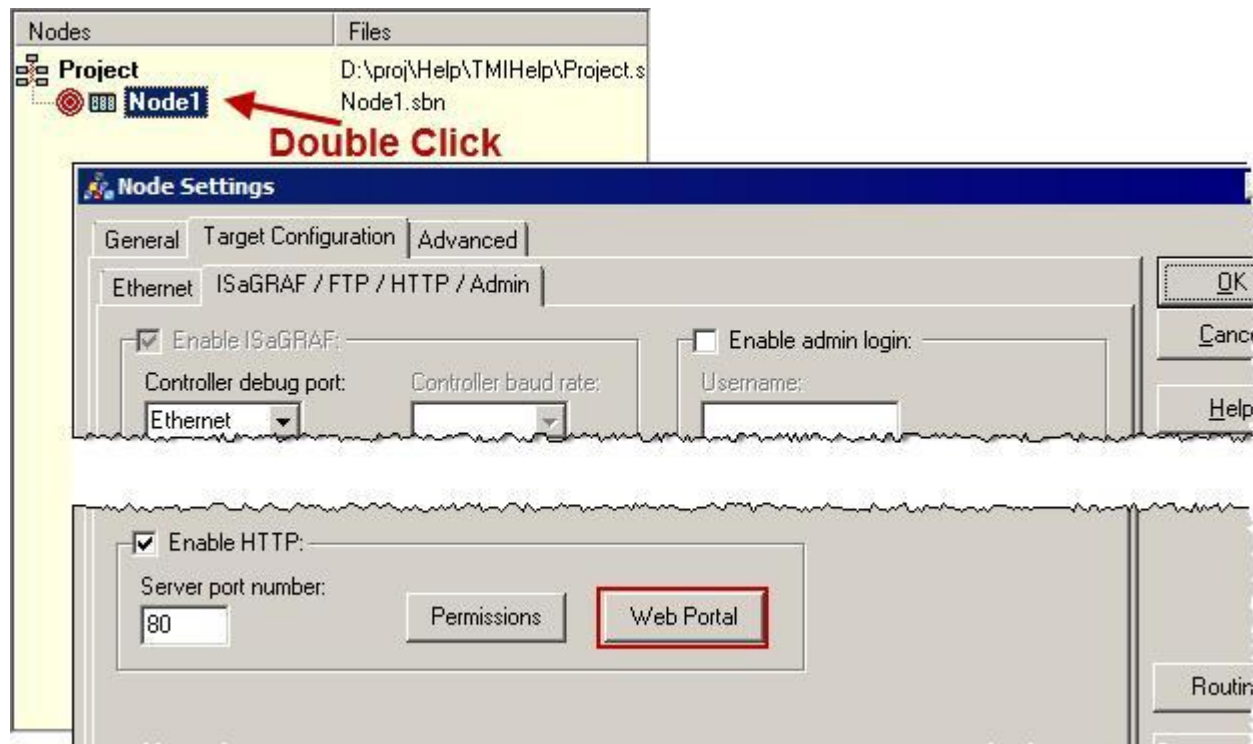
Setting Up an Administrator Account.....	423
Live View -- View and Edit Registers	442
Trend Graphs.....	446
Alarms.....	448
Documents.....	452

Setting Up an Administrator Account

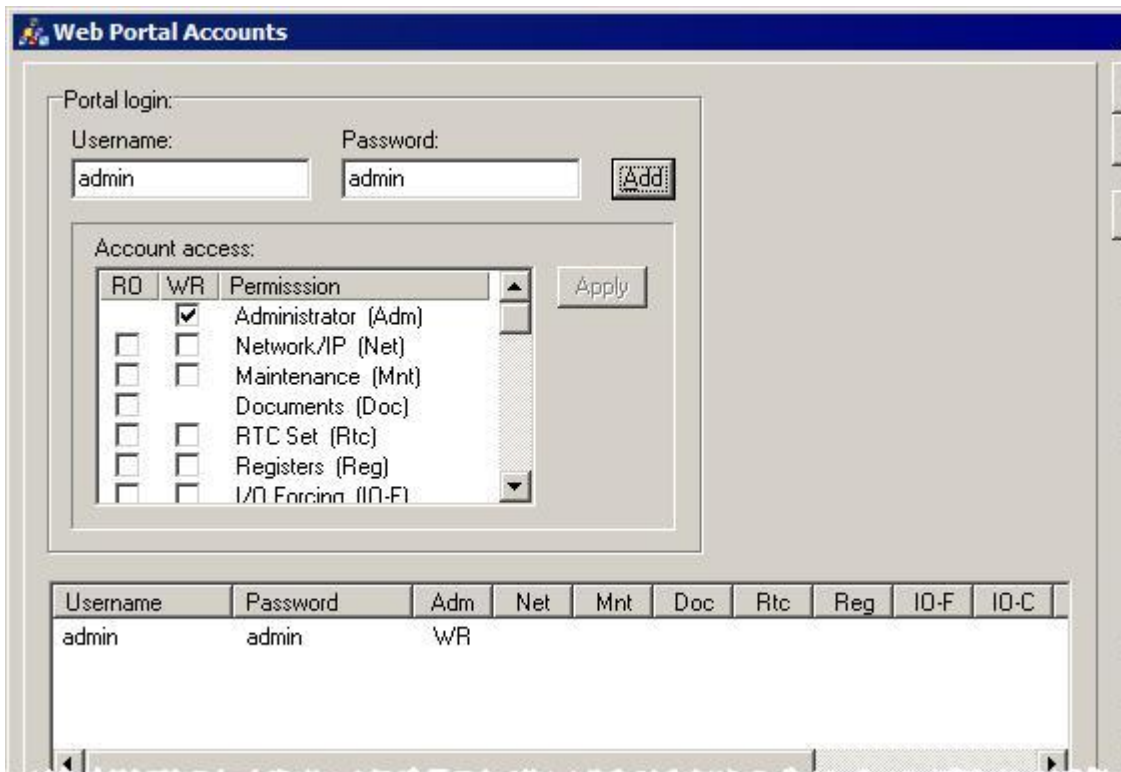
In ScadaBuilder, go to the **Node | Settings** menu or double click on the node of interest in the left hand side of ScadaBuilder.

Go to the Target Configuration | ISaGRAF/FTP/HTTP/Admin tab

Click on the Web Portal button.



The first requirement is to setup an administrator account. Enter a Username and Password (admin and admin in this case) and give this account at least Administrator (Adm) privileges. You can give the account other privileges as well but the administrator is the most important as all other user accounts will be created from this one. Click the Add button when done. You can create other accounts as well here. Once downloaded to the controller, all the accounts here are unchangeable except from the ScadaBuilder software.



The image shows a web browser window titled "Web Portal Accounts". It contains a "Portal login:" section with "Username:" and "Password:" fields, both containing the text "admin", and an "Add" button. Below this is an "Account access:" section with a table of permissions and an "Apply" button.

RO	WR	Permission
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Administrator (Adm)
<input type="checkbox"/>	<input type="checkbox"/>	Network/IP (Net)
<input type="checkbox"/>	<input type="checkbox"/>	Maintenance (Mnt)
<input type="checkbox"/>	<input type="checkbox"/>	Documents (Doc)
<input type="checkbox"/>	<input type="checkbox"/>	RTC Set (Rtc)
<input type="checkbox"/>	<input type="checkbox"/>	Registers (Reg)
<input type="checkbox"/>	<input type="checkbox"/>	I/O Forcing (IO-F)

At the bottom of the window is a table showing the current configuration for the "admin" user.

Username	Password	Adm	Net	Mnt	Doc	Rtc	Reg	IO-F	IO-C
admin	admin	WR							

Click Okay.

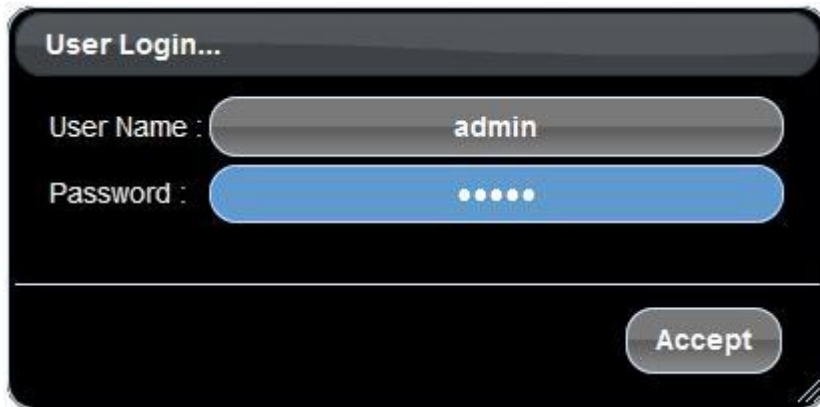
If you have already done a **Target | Complete Controller Setup...** (see "Node Configuration - Downloading" on page 30) then you can click on the Apply To Target button and then the Restart Target button.

If not, Click OK and do a **Target | Complete Controller Setup...** (see "Node Configuration - Downloading" on page 30) now. Once this is done, the Administrator account may be used from the User Portal Interface.

Accessing the User Portal for Administration

Open a web browser and in the address bar, enter the IP address of your controller.

You should see the initial login screen. Enter your new Administrator account credentials.



The 'User Login...' dialog box has a dark background. It contains two input fields: 'User Name' with the text 'admin' and 'Password' with five dots. A blue 'Accept' button is located at the bottom right.

Depending on what permissions you configured for the Administration account different buttons will show in this dialog. For now, we will concentrate on the tasks that only the Administrator can do.

Click on the Administration button to access the the dialog.



The 'User Portal Main Menu' dialog box has a dark background. It features a large blue 'Administration' button at the top. Below it, separated by a horizontal line, are two smaller buttons: 'About' and 'Log Out'.

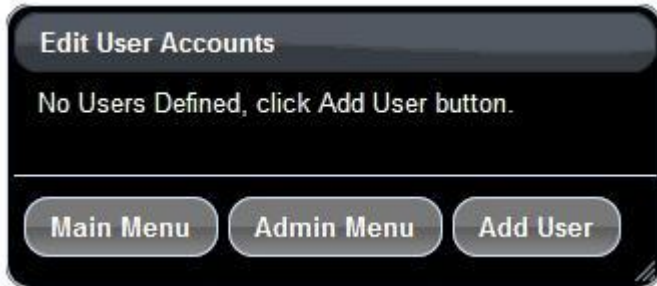
Click the Add/Edit User Accounts button to create a new account.



The 'Administration Menu...' dialog box has a dark background. It contains three stacked buttons: 'Add/Edit User Accounts' (highlighted in blue), 'Add or Edit Links', and 'Change Password'. A 'Main Menu' button is located at the bottom right.

Adding a New User Account

In the Administration | User Account dialog click Add User



Enter a User Name and Password. Enter the password again in the Verify field.

Select the user permissions from the drop down lists and click the Apply button when finished.

A screenshot of a dark-themed dialog box titled "Add New User Account". It contains several input fields and dropdown menus. The "User Name" field contains "user1". The "Password" and "Verify" fields are masked with dots. Below these are 14 dropdown menus for permissions: Administrator, Network/IP, Maintenance, Documents, RTC Set, Registers, Gas Setup, Gas Calibration, Trending, Alarms, Tui 1, Text Message 1, and Maint. Each dropdown menu has a selected value (Write, Read, or Access). At the bottom, there are four buttons: "Main Menu", "Admin Menu", "Cancel", and "Apply".

Field	Value
User Name	user1
Password
Verify
Administrator	Write
Network/IP	Write
Maintenance	Write
Documents	Read
RTC Set	Write
Registers	Write
Gas Setup	Write
Gas Calibration	Write
Trending	Read
Alarms	Write
Tui 1	Access
Text Message 1	Access
Maint	Access

User Portal Permissions

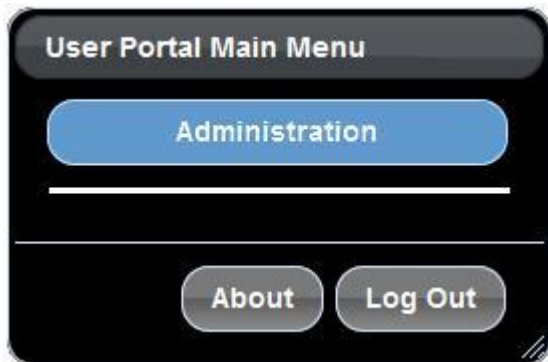
Permission	Access	Description
Administrator	Read/Write	Allows users with this permission to create and edit other user accounts as well as adding Web/HMI links to the interface for user account access.
Network/IP	Read/Write	Allows users to set and change the IP Address, Network Mask, Gateway, and DNS Server addresses of the controller.
Maintenance	Read/Write	Allows users to turn on Radio Diagnostics for internal radio options on the controllers.
RTC Set	Read/Write	Allows users to synchronize the Real Time Clock with the local PC's. If either Read or Write permission is enabled, the Real Time Clock will be checked any time the Main Menu of the User Portal is displayed. User will be prompted if the controller's clock is off from the PC's clock.
Alarms	Read/Write	Alarms that are assigned to Alarm Groups in ScadaBuilder will be accessible from the Alarms button. By default an alarm task will monitor the state of these alarms and pop-up a dialog should any alarms become active. Users with Write permissions may choose to acknowledge the alarm at that time or do so from the Alarms dialog button in the Main Menu. If the Autolog option is checked for the alarms, users with Read or Write permissions can query date ranges and generate and download alarm reports.
Documentation	Read	Documents that are downloaded to the WebRoot\docs directory either from the ScadaBuilder Node Documents configuration or manually through an FTP client will be accessible from the Node Documents... menu.
Gas Setup	Read/Write	Allows users to configure any AGA Gas Calc run programmed into the controller.
Gas Calibration	Read/Write	Allows users to calibrate sensors for and AGA Gas Calc run programmed into the controller.
Trending	No Access/Access	Allows users to access any Trends on the controller from the front panel of the User Portal
Any TUI	No Access/Access	Allows users to access a TUI to a security level from the TUI login.
Any TMI	Read/Write	Allows the user to access the Text Message Interface if configured from an outside cell phone by text message.

Any Link	No Access/Access	Allows users to access Links from the Web/HMI links interface of the front panel of the User Portal. Web/HMI Links must be created using an account with Administration privileges before they will be available for User account configuration.
----------	------------------	--

Only accounts with administrator permissions may modify user accounts other than users changing their own password.

Editing and Deleting User Accounts

To edit an existing user account, log in as Administrator and click on the Administration button.

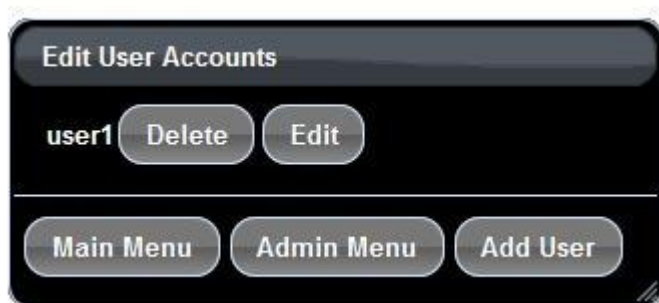


Editing a User Account

Click the User Accounts button to edit an existing account.



Click the Edit button.



Here all of the permissions available may be assigned or removed from the current user. Select the permissions you want or if a permission is no longer allowed, set that permission to None or No Access.

Edit User Account

Password

Verify

Administrator Write

Network/IP Write

Maintenance Write

Documents Read

RTC Set Write

Registers Write

Gas Setup Write

Gas Calibration Write

Trending Read

Alarms Write

Tui 1 Access

Text Message 1 Access

Maint Access

Main Menu Admin Menu Cancel Apply

If no password is entered in the Password and Verify field, then it is ignored and the user's password remains the same. If a new password is assigned, the Password and Verify fields must match exactly. Spaces and special characters are not allowed in the password itself. Click apply when done to get back to the User Account menu.

Deleting a User Account

Edit User Accounts

user1 Delete Edit

Main Menu Admin Menu Add User

Click on the Delete button and the User Account will be deleted from the controller.

Adding and Editing Web/HMI Links

To add a link to the interface that other users can access, you must first be logged in with an Administrator account.

From the Main Menu, click on the Administration button.

Click on Add or Edit Links.



Click on the Add Link button.



Enter a Web/HMI Link Name. This is the name that will show up when assigning User Permissions to it and also the link name that will show up in the Links dialog accessed from the Main Menu.



Enter a the HTTP link in the Link: fields. There some rules to this that follow other URL shortcut interfaces.

If you are accessing a site that is on the controller, it should start with a forward slash "/".

The path must include the entire path name for any URL including the file name extension (".html" as in the example above).

Paths for links are relative to the WebRoot directory so any Hibeam page may be linked as in the above example.

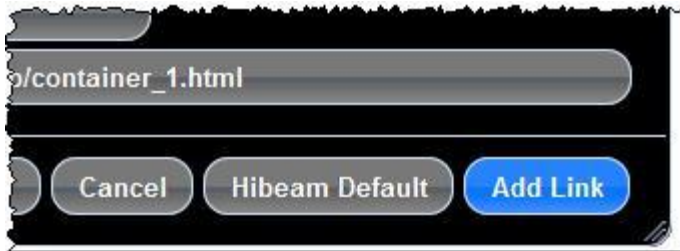
Paths that are not on the controller should be prefixed with `http:<ip address>/<path(s)>/<filename>.<extensions>`

Some examples are:

```
http://demo.iclinks.com/
http://71.14.140.233/
http://demo.iclinks.com/login.html
```

If the filename and extension are left off of the path the a trailing forward slash should be used to get to a website's default html file (usually `index.html` or `index.htm`) as in the first and second example above.

Click the Add Link button to complete the operation.



Add as many Web/HMI Links as are necessary for your system and then assign their permissions to each user as appropriate.

You can Test the Web/HMI Link by clicking the Test button.



A new browser window or new browser tab will start (depending on your browser and its configuration).

If the Web/HMI Link is properly entered you will see the expected web page. If not, click the edit button and try again.

Any account with administration privileges can add, edit and delete links.

If a link is deleted, it will disappear from the Main Menu | Web/HMI Links interface.

Adding and Accessing Web/HMI Links Via a User Account

To add a Web/HMI Link to a user account, login in to an Administrator account and click the Administration button. Add a user or edit and existing user and select Access for any links the user should have access to.

Links	Access
Tui 1	Access
Text Message 1	Access
Maint	Access
HiView	No Access

Click the Apply button when done.

To access the link from the user's page, return to the Main Menu and Click the Log Out button.

Login as the user.

User Name :	Password :
user1

Click on the Web/HMI button.



If there is only one link defined for the user (or in the system) then the page will load automatically. When more than one link is defined for a user the Web/HMI Links you configured for this user should be displayed in the dialog.



Click on the Link button to access the web page.

A new browser window or new browser tab will start (depending on your browser and its configuration).

You should see the page as you configured it.

Changing The IP Settings

To change the IP Settings from the User Portal, a user must have Network/IP permissions set to Write.

From the Main Menu, click on the Administration Button then click on the Network button.



You should see a dialog that will allow the changing of the IP address, Network Mask, Gateway and DNS servers.

If the IP address has never been changed from the User Portal then the default IP Settings that are configured in **ScadaBuilder Node | Settings** will be shown.

The image shows a 'Controller IP Address' configuration dialog. It features a table for setting IP parameters and three buttons at the bottom.

	192	168	237	210
IP Address	192	168	237	210
Network Mask	255	255	255	0
Gateway Address	0	0	0	0
DNS Server 1	0	0	0	0
DNS Server 2	0	0	0	0

Buttons at the bottom: Main Menu, Admin Menu, Write IP Setup

These settings are stored in separate file on the controller in the root directory as <node name>.net. Deleting this file will reset the controller back to those defaults.

To change the IP Settings, simply type the new settings into the appropriate text boxes and click the Write IP Setup button.

Controller IP Address				
IP Address	192	168	237	213
Network Mask	255	255	255	0
Gateway Address	192	168	237	254
DNS Server 1	192	168	237	1
DNS Server 2	192	168	237	2

[Main Menu](#) [Admin Menu](#) [Write IP Setup](#)

If the IP settings are coherent, they will be stored to the controller. To make the settings take affect, you must log out of the User Portal (Main Menu) or restart the controller. This way, any other administrator tasks can be finished on the current IP address.

Click the Cancel button to abort writing the values.

See *Appendix A, An Ethernet/Internet Primer for TCP/IP* (on page 649) for definitions of each IP configuration and how they are used.

Setting Radio Diagnostic Mode

To facilitate setting up an internal radio option on the controller, the diagnostic switch to access the radio diagnostics through a serial terminal has been implemented in the User Portal interface. This was previously done from the ScadaBuilder Target | Radio | Freewave/MDS or Digi | Enable Diagnostics menu. For more details on radio option configuration, see the Pinnacle Technical Reference manual available on the ICL website. The Communications Options section of the document has more detailed information on radio configuration serial connections, tools and settings.

To use the Radio Configuration option, a user account must be logged in with Maintenance Write permissions.

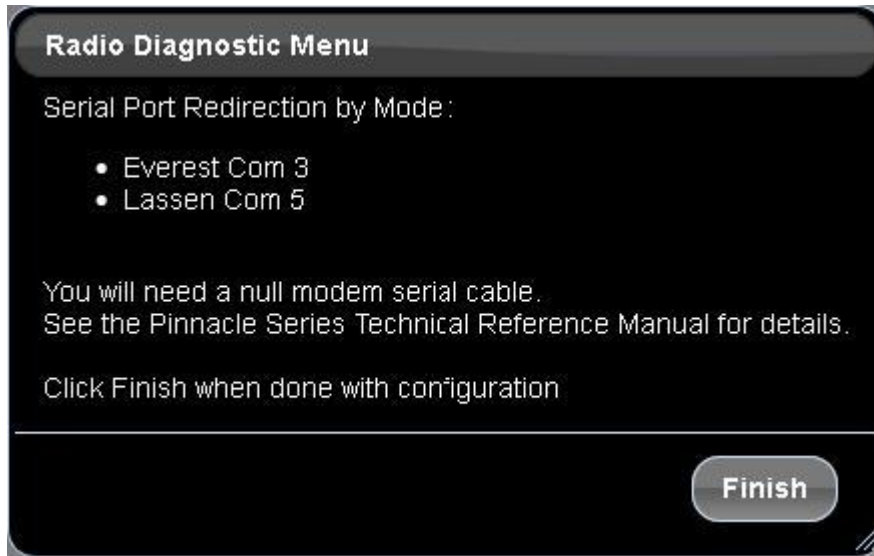
To turn on the Radio Configuration from the Main Menu click on the Administration button then click on Radio Configuration.



If your controller has a Digi/MaxStream internal radio, click on the Digi_MaxStream button.



If your controller is configured for a Radio option in the **Node | Settings** then this dialog will come up. Otherwise an error will be displayed.



If your controller has a Freewave or MDS internal radio, click on the Freewave button (both radios use the same mode on all controllers). You should see the dialog above.

For configuration details, see the Communication Options section of the Pinnacle Technical Reference manual.

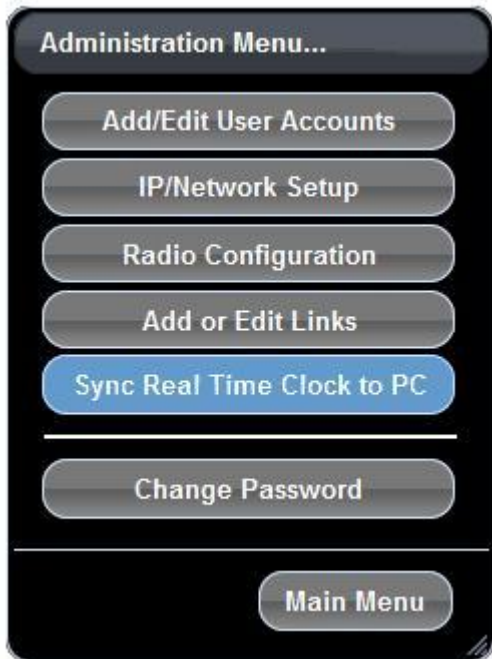
While this dialog is open, all Network Sessions using the Internal Radio port will be disabled and the diagnostic information is redirected to the ports shown above. The mode will continue until the Finish button is clicked allowing for serial access to the radio's diagnostic port.

When you are done configuring the radio (or accessing its internal diagnostics), click the Finish button.

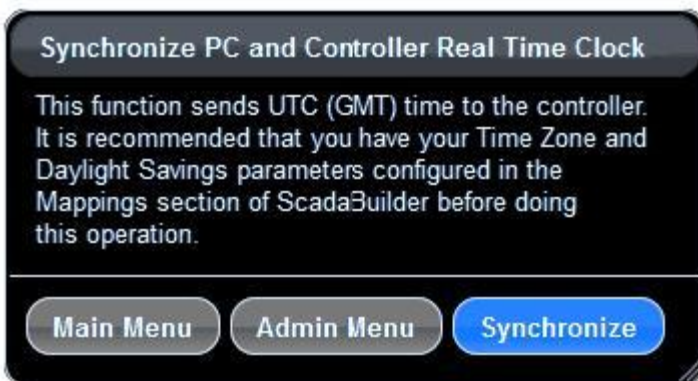
Synchronizing The Real Time Clock

To do this task, a user must have RTC Write permissions.

In the Administration Menu, click on the Sync Real Time Clock to PC button.



To synchronize the clock of the controller to the clock on you PC click the Synchronize button.



If the command was successful then you will see this dialog:



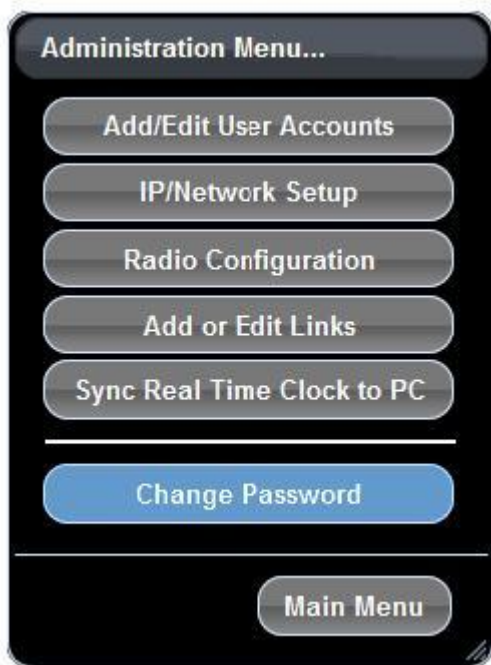
It is recommended to do this periodically to keep Trend files in sync (which use the controller's RTC to time stamp data). Be sure that your PC's clock is correct and that the Time Zone and Daylight Savings Time on your PC are correctly setup.

If a user has RTC Write or RTC Read permissions, the clock will be checked when the Main Menu is first loaded. If the clock is off by more than a minute, the user will be prompted to change it (Write permission) or warned that it is off and by how much (Read permission).

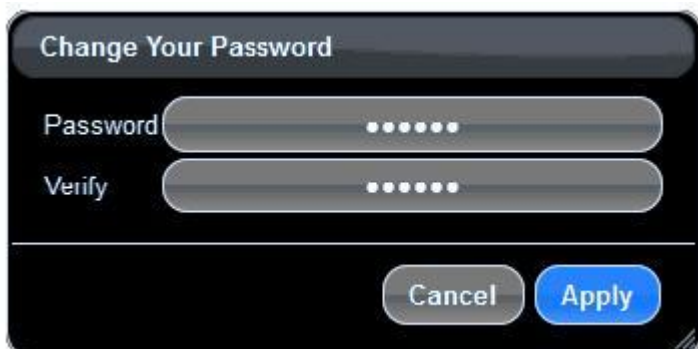
Changing Your Password

Any user can change their own password.

In the Administration dialog (click the Administration button from the Main Menu) click the Change Password button.



Enter the new password and click Apply.



Only alpha-numeric characters are allowed. Passwords are case sensitive.

If a password is forgotten, then it must be restored by a user with Administrator Write permissions.

See *Editing and Deleting User Accounts* (see "Editing and Deleting User Accounts" on page 429) for details.

Live View -- View and Edit Registers

Reading registers from the portal requires that the user account have at least Registers Read permissions set.

If the user needs to write registers then Register Write permissions must be set on the account.

When the application is configured, registers that are exposed to the user must be setup and have a minimum requirement of having the symbol permission set, a register index assigned and a name that will be recognizable by the user when accessing the register in the User Portal. These configurations are done in ScadaBuilder at programming time.

Register Allocation - Reals

Register names:

Prefix: Enum: Template file: Enum: Suffix:

Sample output: Index: Count:

Name	Index	Value	Retained	Symbol	Co
GFDOut_Vol_Rate	2098		None	Read	
StatPresLow	1	50.0	File	Write	
StatPresHigh	2	60.0	File	Temp	

Name = name the user will see in the Portal.

Index = necessary for access.

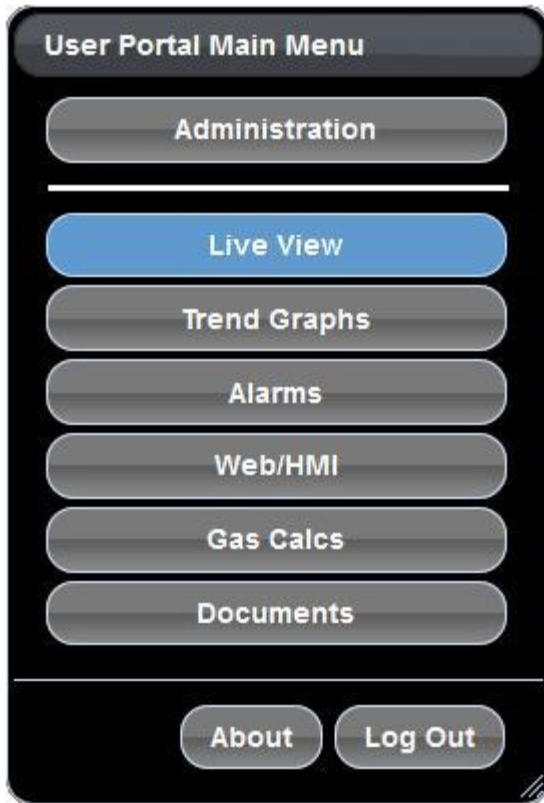
Symbol = access type including

Read = read only

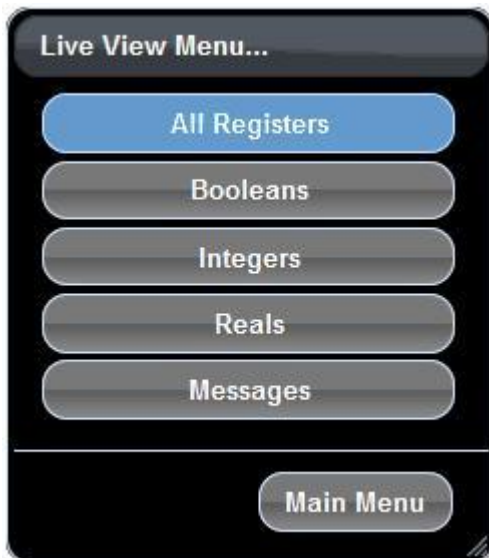
Write = read write

Temp = temporarily changeable that can be Aborted or Confirmed.

Once these changes are made to the Register Allocation, the application is downloaded to the controller (Lightning Bolt button) and the user permissions setup appropriately as shown above, after login, the Main Menu will show the Live View menu option. Click on it to get to the Live View menu.



Click on the data type of interest or All Registers to show all registers configured.

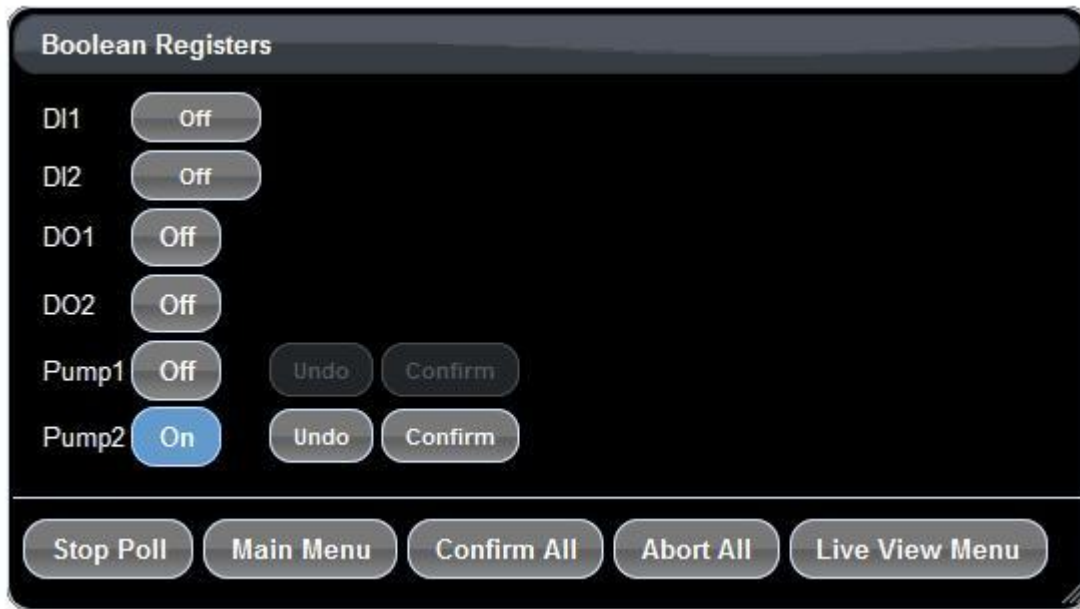


If a data type is not configured in the Symbols column in ScadaBuilder, that data type will not show up here.

Boolean Registers

Click on the Boolean Registers button in the Main menu

For booleans the following data types are shown like this.



To change a read/write or temporary boolean, simply click on the button.

Booleans setup for temporary writes have an Undo and Confirm button and the Confirm All and Abort All buttons show whenever the Temporary registers are written to. The Undo and Confirm on the register will operate on only the register at that line where as the Confirm and Abort All will operate on all registers that have been changed in this login's session. If a Temporary variable has not been written to, the Undo and Confirm buttons for that register's line are disabled.

If Temporary registers are left in the unconfirmed state, then they will change back when the controller is reset.

Integer and Real Registers

Click on the Integer or Reals button in the Main menu

For Integers and Reals the following data types are shown like this.

The screenshot shows a web interface titled "Real Registers". It contains a list of five registers, each with a text input field showing its current value. The registers are:

- Tank1: 32.8
- Tank2: 6.28
- spTank2LowAlarm: 22.8 (This field is highlighted in blue, indicating it is currently being edited. To its right are "Undo" and "Confirm" buttons.)
- Input_Voltage: 222.61
- spHighCurrentCutoff: 12.6 (This field is also highlighted in blue. To its right is an "Undo" button.)

At the bottom of the interface, there is a row of five buttons: "Stop Poll", "Main Menu", "Confirm All", "Abort All", and "Live View Menu".

To change a read/write or temporary registers, simply click on the register and type in the number. You can either hit enter, click on another register or press tab to move to the next field to write the value. During editing, the polling cycle (about a one second interval) will not write the register being edited for about 10 seconds unless a key is pressed. After 10 seconds of inactivity, the field will revert back to polling mode again.

Integers and Reals setup for temporary writes have Undo and Confirm buttons and the Confirm All and Abort All buttons show whenever the Temporary registers are written to. The Undo and Confirm on the register will operate on only the register at that line where as the Confirm All and Abort All will operate on all registers that have been changed in this session. If a Temporary variable has not been written to, the Undo and Confirm buttons for that register's line are disabled.

If Temporary registers are left in the unconfirmed state, then they will change back when the controller is reset.

Message Registers

Click on the Messages button in the Main menu

For Message registers the following data types are shown like this.



The screenshot displays a window titled "Message Registers" with a dark background. It contains three rows of data, each representing a message register. Each row has a label on the left, a text input field in the middle, and one or two buttons on the right. The first row is labeled "Phone_Number_1" and contains the value "1503-728-1677" with an "Undo" button. The second row is labeled "Phone_Number_2" and contains the value "1503-642-1547" with "Undo" and "Confirm" buttons. The third row is labeled "Phone_Number_3" and contains the value "1503-652-1709" with an "Undo" button. At the bottom of the window, there is a horizontal bar with five buttons: "Stop Poll", "Main Menu", "Confirm All", "Abort All", and "Live View Menu".

Register Label	Value	Buttons
Phone_Number_1	1503-728-1677	Undo
Phone_Number_2	1503-642-1547	Undo, Confirm
Phone_Number_3	1503-652-1709	Undo

Bottom Bar Buttons: Stop Poll, Main Menu, Confirm All, Abort All, Live View Menu

To change a read/write or temporary Message registers, simply click on the register and type in the string. You can either hit enter, click on another register or press tab to move to the next field to write the value. During editing, the polling cycle (about a one second interval) will not write the register being edited for about 10 seconds unless a key is pressed. After 10 seconds of inactivity, the field will revert back to polling mode again.

Message registers setup for temporary writes have an Undo and Confirm button and the Confirm All and Abort All buttons show whenever the Temporary registers are written to. The Undo and Confirm on the register will operate on only the register at that line whereas the Confirm All and Abort All will operate on all registers that have been changed in this session. If a Temporary variable has not been written to, the Undo and Confirm buttons for that register's line are disabled.

If Temporary registers are left in the unconfirmed state, then they will change back when the controller is reset.

Trend Graphs

To access Trend, a user must have Trend Access permissions.

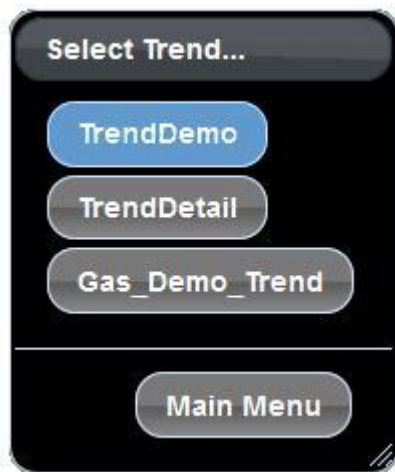
To configure one or more Trends, see the *Trending* (on page 455) section.

To access Trends, click the Trend Graphs button.



If there is only one Trend configured, it will be brought up in its own browser tab or window depending on your browser's configuration. You will need to allow access to popups in most browsers for this to work.

If more than one Trend is configured, a list of buttons showing each Trend name will come up in a dialog.



Select the Trend of interest. You can bring up more than one Trend as each is treated as its own web page and leave them up if you like—even if you shut down the User Portal window.

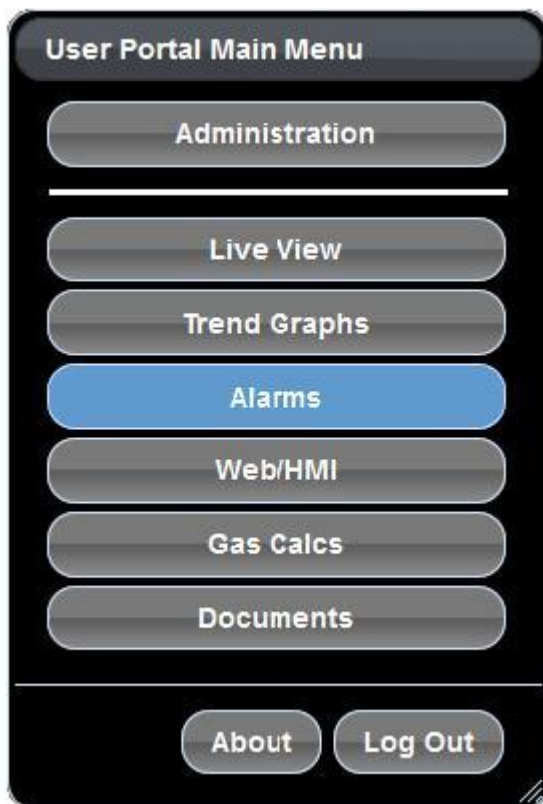
Alarms

To use Alarms in the User Portal, a user must have either Alarms Write or Alarms Read permissions. If the user has no permissions for Alarms, then the button will not show in the Main Menu. If no Alarms are configured in the system then it will pop up and say that there are no Alarms.

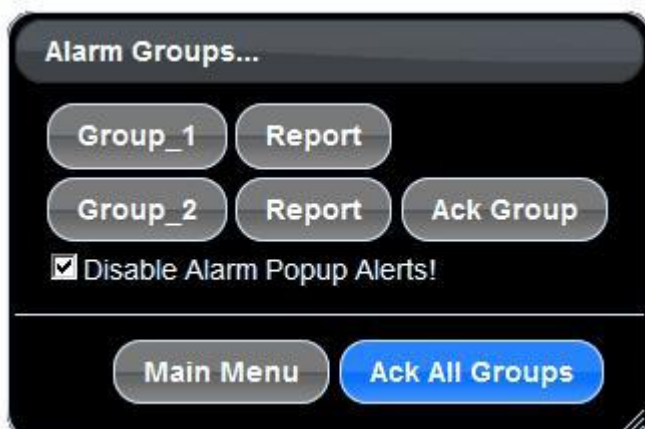
To configure Alarms for your system see the *Using Alarms* (on page 159) section.

Users with Alarm Read access can look at Alarm and Alarm Group States but cannot acknowledge them--they can also generate Alarm Reports for those Alarms that have the Autolog feature turned on. See *Alarm Options* (on page 161) for more details.

To access Alarms, click on the Alarms button in the Main Menu.



This brings up the Alarm Groups... dialog.



Groups give you a way to organize your Alarms, Generate Alarm Reports and acknowledge Alarms.

From this dialog, you can Acknowledge All Groups or Acknowledge an individual Group.

You can also generate Alarm Group Reports (includes all Alarms in that Group).

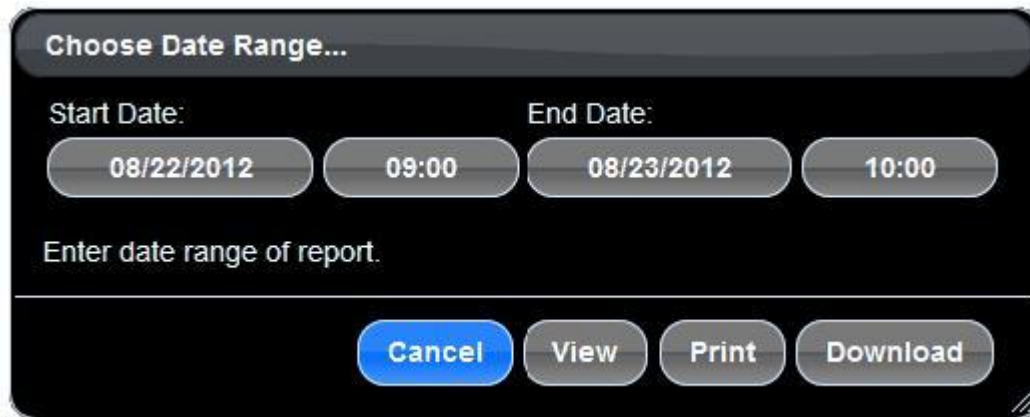
See *Generating Alarm and Alarm Group Reports* (on page 449) for more details.

If you want acknowledge all Alarm Groups, click the Ack All Groups button.

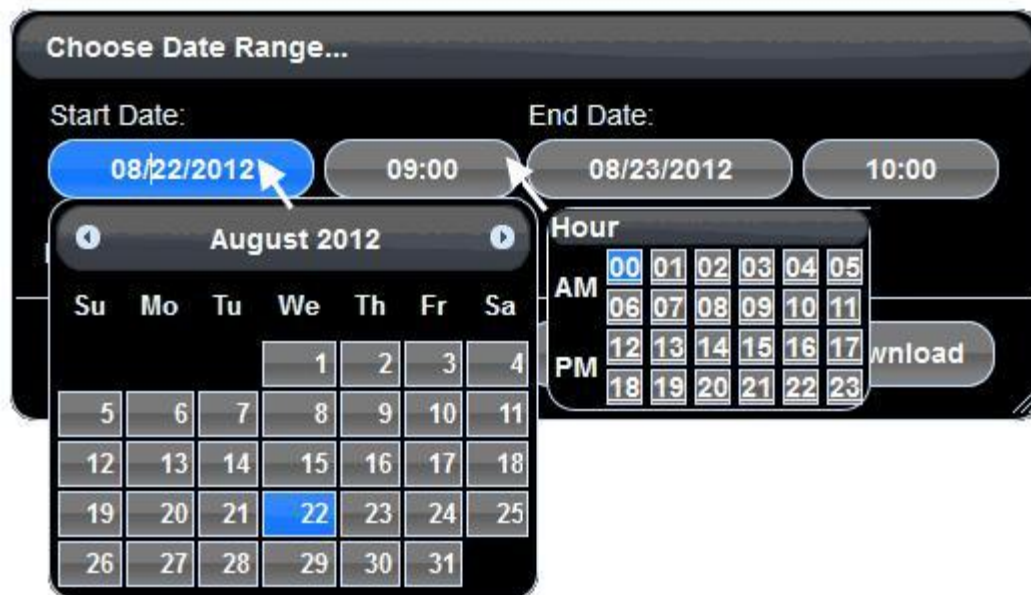
If you want to suppress the Active Alarms! popup, check the Disable Alarm Popup Alerts! See *Active Alarms! Popup* (on page 451) section for details.

Generating Alarm and Alarm Group Reports

After clicking on an Alarm Group Report button or an Alarm Report button, you will see this Choose Date Range... dialog:

A screenshot of a web-based dialog box titled "Choose Date Range...". The dialog has a dark background with light-colored text and buttons. It contains two rows of date and time selection fields. The first row is labeled "Start Date:" and "End Date:". Below these labels are four buttons: "08/22/2012", "09:00", "08/23/2012", and "10:00". Below these buttons is a text input field labeled "Enter date range of report.". At the bottom of the dialog are four buttons: "Cancel", "View", "Print", and "Download". The "Cancel" button is highlighted in blue, while the others are grey.

The default is to get the last 24 hours of data for your report. You can change this date range by clicking on any of the date or time fields and selecting from the calendar and hour interface to get the data of interest:



Once the date range is selected, then you can choose to View, Print or Download the report.

If you select View, you will get a new HTML page with color coded data.

If you select Print, you will get a printable HTML page (black and white) and prompted to select a printer.

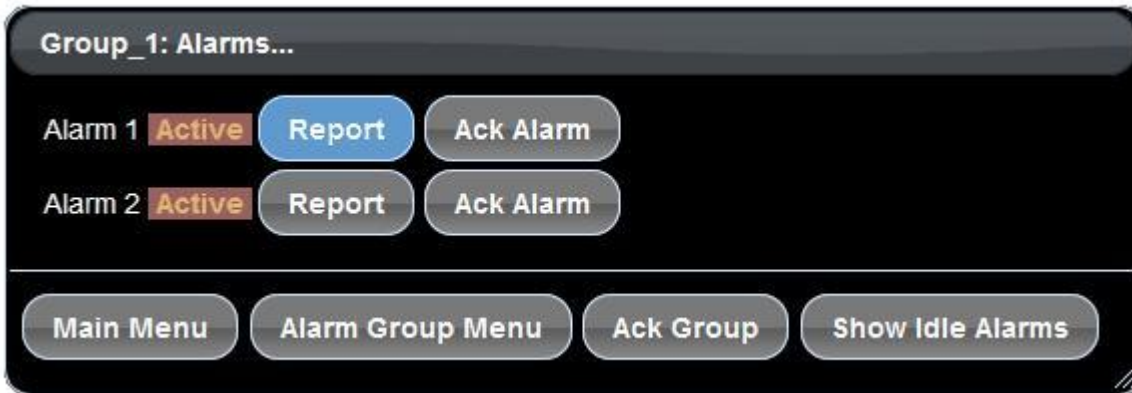
If you select Download, you will get a prompt to save a Report.CSV file.

Accessing Alarms in Alarm Groups

To look at individual Alarm select the group from the right hand column of buttons.



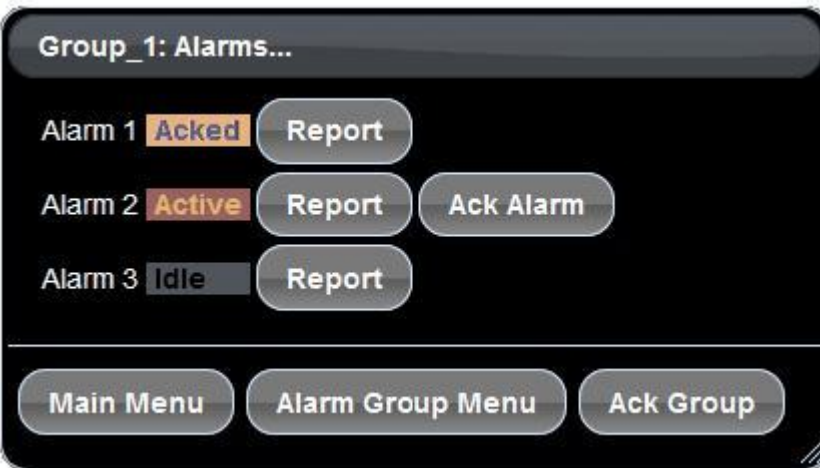
This will give you the <Group>: Alarms... dialog.



This will show all Active and Acknowledged Alarms. From here you acknowledge an Alarm by clicking on the Ack Alarm button. The dialog will be updated to reflect the new Alarm state.

If you want to acknowledge all Alarms on the screen, you can click the Ack Group button.

If you want to view all Alarms in the group, click the Show Idle Alarms:



The Report button will show up only if the AutoLog option is turned on in the Alarm configuration in ScadaBuilder. See *Alarm Options* (on page 161) for more details.

Click on the Report button to get the Alarm history for an individual Alarm.

See *Generating Alarm and Alarm Group Reports* (on page 449) for more information.

Active Alarms! Popup

Another feature of the User Portal is the Active Alarms! Popup. While logged into the User Port, users with Alarm Read and Alarm Write Access can be alerted when one or more Alarms become active.



Users with Alarm Write access can acknowledge these Alarm from the dialog by clicking the Ack All Groups.

Users with Alarm Read access will not see the Ack All Groups button.

The Active Alarms! popup can be disabled temporarily for 24 hours by clicking the Disable Alarm Popup Alerts! check box. The check box is stored to a cookie so it is only for the current user in the current browser and does not affect other users and other browser clients (or on other devices).

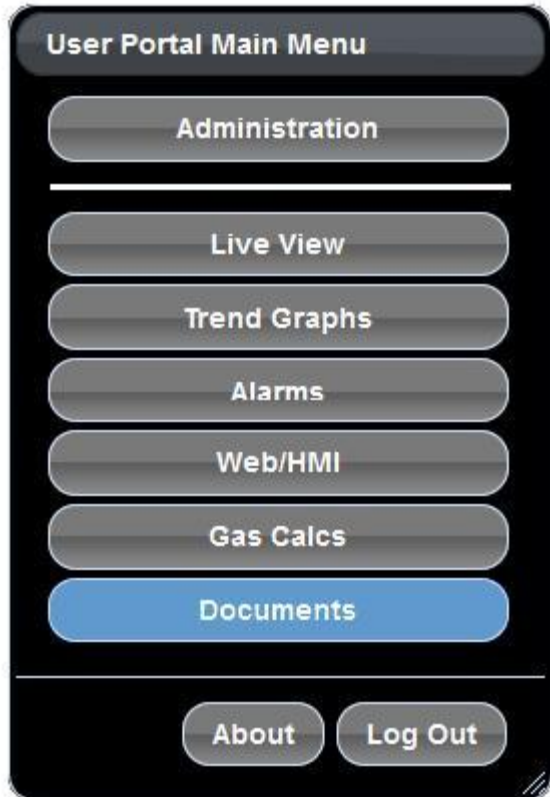
This disable can also be checked from the Alarm Groups Menu. When a user is in the Alarms area looking at Alarm Groups or individual Alarms, the Active Alarms! popup is disabled.

The only other place that the Active Alarms! popup is suppressed is during an active AGA Gas Calibration procedure.

Documents

To view Documents loaded on the controller a user must have the Documents Read permission.

In the Main Menu, click on the Documents button.



If no documents are loaded on the controller, the user will be alerted and the Main Menu dialog will be reloaded.

If only one document is available, an attempt to load it will happen automatically.

If more than one document is available, the Documents dialog will appear with a list of documents on buttons to select.



Simply click on the document of choice.

The document will be automatically downloaded to whatever viewer is available in the browser's plugins. If no plugin is found, then the user will be asked to save the file.

To download documents to the controller automatically with a Complete Controller Setup... see ***NODE Menu*** (on page 56) and the Documents... section.

Documents are stored on the IDE drive under c:\WebRoot\docs. Any files in that directory will show in the above list.

SECTION XX

Trending

Trending utilizes three basic components of ICL Pinnacle series controllers:

- Standard 512MB IDE Flash disk
- High Performance Web Server
- Ability to recall data from a CGI interface

Trending allows the user to operate in two basic modes

Strip Chart mode that emulates a paper strip chart to be updated at a user defined interval.

Historical mode that allows a user from the web interface to recall and display older archival data stored by the trending system.

Here is an example of what a trend can look like (although many different looks may be used even on the same controller):



The look and feel can be changed by applying different, themes, trace colors, trace styles and legend selections.

In This Section

Setting Up a New Trend	456
Accessing a Trend Web Page	466
Reports Button	467
Traces Button.....	470
Settings Button.....	471
Date Range Button	474
Info Button	476
Stats Button	477
Refresh Button	478
Mode (Strip/History Selector)	478
Zooming (PC Browsers)	479
Cursor: Mode Hovering and Zooming With iPad and iPhone Browsers	481

Setting Up a New Trend

There are several tasks to do before setting up a new trend.

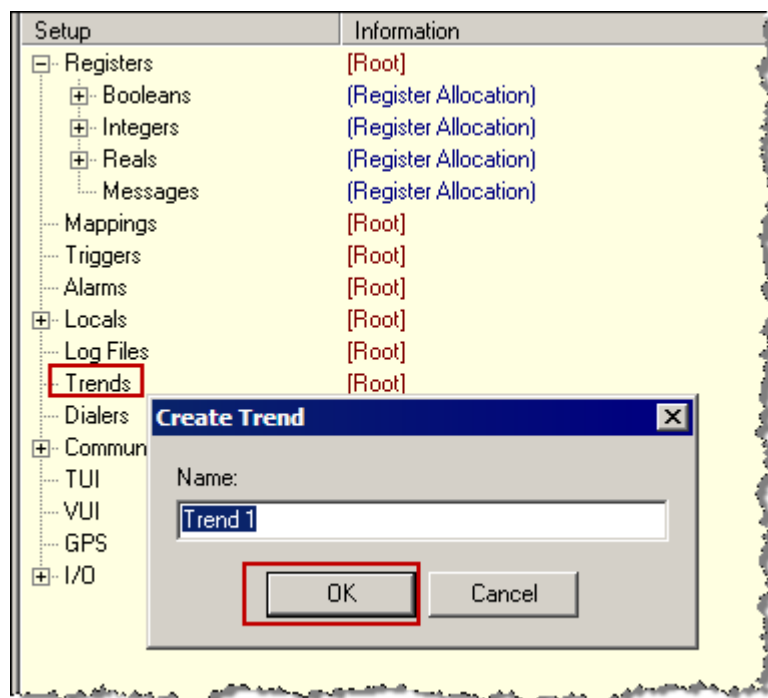
- 1) Map your timezone in the **Mappings** (see "Mappings Reference" on page 493) Section for your local time zone. You can do this to a retained integer if you wish to configure the time zone differently once installed.
- 2) Map your Day Light Savings flag to a register (make it retained) in the **Mappings** (see "Mappings Reference" on page 493) and set the flag to the proper value.
- 3) Download the application to the target controller.
- 4) Got to the **Target | Synchronize RTC** menu option and synchronize the controller's Real Time Clock.

Now you are ready to build your trend.



If you fail to do these steps, the Trend data may not show up immediately due to clock to PC synchronization errors.

Double click on the Trends section in the Setup window in ScadaBuilder



You can use the default names or put in your own. We recommend you do not use spaces in the names as they are confusing when accessing the trend in the browser. The spaces in the name are replaced with an underscore in the URL when accessing a Trend in a web browser. See **Accessing a Trend Web Page**. (see "Accessing a Trend Web Page" on page 466) Trends can also be accessed from the User Portal web interface. See **User Portal (Web Interface)** (on page 423) for details on how to set this up.

Click OK and you should see the following Dialog:

Trends - Trend 1 [?] [X]

General | **Traces**

Parameter	Value
Data Configuration	
Data sample rate (sec)	60
Data days map (days)	(none)
Data days limit (days)	30
Data bytes limit (MB)	10
Data file drive	(default)
Data file directory	
Page Layout	
Page size pixels (W, H)	900 300
Page position (X, Y)	0 0
Page theme	(default)
Page heading	
Page Fonts	
Font name	Arial
Font size	14
Font color	RGB = {0x00, 0x00, 0xA0}
Font style	Normal
Page Options	
Strip chart mode	Enabled
Strip chart duration (hrs)	2
Strip chart refresh (sec)	10
Left axis range (min, max)	0.0 0.0
Right axis range (min, max)	0.0 0.0
Boolean top margin height (%)	10
Trace line width	1
Trace shadow depth	2
Toolbar background color	RGB = {0xD5, 0xE5, 0xF1}
Gradient start color	RGB = {0xFC, 0xFD, 0xFD}
Gradient end color	RGB = {0x6D, 0xA6, 0xD1}
Averaging decimal places	3
Legend display	Enabled
Statistics range display	Enabled
Statistics average display	Enabled
Cursor hover display	Enabled

[OK] [Cancel] [New] [Copy] [Delete] [Notes] [Help]

[Previous] [Previous] [Next] [Next] [Rename] Trend 1 [Dropdown] 1 of 1

If you want to use the defaults for the trend, simply go to the Traces section to add traces.

Trend Parameters

Parameter	Value
Data Configuration	
Data sample rate (sec)	60
Data days map (days)	(none)
Data days limit (days)	30
Data bytes limit (MB)	10
Data file drive	(default)
Data file directory	
Page Layout	
Page size pixels (W, H)	900 300
Page position (X, Y)	0 0
Page theme	(default)
Page heading	
Page Fonts	
Font name	Arial
Font size	14
Font color	RGB = {0x00, 0x00, 0xA0}
Font style	Normal
Page Options	
Strip chart mode	Enabled
Strip chart duration (hrs)	2
Strip chart refresh (sec)	10
Left axis range (min, max)	0.0 0.0
Right axis range (min, max)	0.0 0.0
Boolean top margin height (%)	10
Trace line width	1
Trace shadow depth	2
Toolbar background color	RGB = {0xD5, 0xE5, 0xF1}
Gradient start color	RGB = {0xFC, 0xFD, 0xFD}
Gradient end color	RGB = {0x6D, 0xA6, 0xD1}
Averaging decimal places	3
Legend display	Enabled
Statistics range display	Enabled
Statistics average display	Enabled
Cursor hover display	Enabled

Navigation: Previous, Next, First, Last, Rename, Trend 1, 1 of 1

Data Days Register Map

Allows the user to map an integer register that shows how many days worth of data are currently stored on the controller for this Trend.

Data Sample Rate (Sec)

Sets the default sample rate for all the traces in this trend. The rate defines how often the trend module looks for change of data on the variables used in the Traces Tab. For efficiency, Trend data is not written to the controller's disk unless a change is detected from the last known sample.

Data Days Limit (days)

Limits the number of days stored on the controller for this Trend.

This parameter works in conjunction with Data Bytes Limit (MB), whichever limit is reached first will limit the data stored.

Limiting data is accomplished by deleting the oldest day file after a limit is reached.

Data Bytes Limit (MB)

Limits the amount of data stored for this Trend in MegaBytes on the disk.

This parameter works in conjunction with Data Days Limit (Days), which ever limit is reached first will limit the data.

Limiting data is accomplished by deleting the oldest day file after a limit is reached.

Data File Drive

The physical drive where the trend data is stored.

Can be set to one of the following:

IDE (default)

USB1

USB2

USB3 (Everest Only)

USB4 (Everest Only)

Data File Directory

Used in conjunction with Data File Drive to allow the storage of Trend data to a subdirectory on a given drive. This has no effect on where the web pages for Trending reside.

Page Size Pixels (x,y)

Trend size on the web page in pixels.

x = width

y = height

Page Position (x,y)

Allows the Trend to be place at an absolute position on the web page.

Page Theme

ScadaWorks has some built in themes that make the trend prettier than just a basic graph.

Included are the

Bw (Black and White theme), Redmond, Cupertino, South Street and Vader themes. Try them out. They can be changed and downloaded with a lightning bolt download of the application. Don't forget to refresh the web page in the browser when you change themes.

Themes are downloaded to the controller with a **Target | Complete Controller Setup...** only when a Trend is configured in the application.

Page Heading

This allow the user to place a title on the page that is displayed above the Trend Graph itself.

Font Name

Any font available to the web browser may be selected here. If the selected font is not found when the Trend is served up in the Browser, the default "Arial" will be used.

Font Color

Red Green and Blue parameters may be specified for the text in the Trend. Affects all text in the Trend except the Button and Text Box Controls which are controlled by the theme.

Font Style

May be set to "Normal" or "Bold". Affects all text in the Trend except the Button and Text Box Controls which are controlled by the theme.

Strip Chart Mode (Enable)

Controls whether the default mode is Strip or History. Can be overridden by the Strip/History Mode: selector in the Trend graph controls.

Strip Chart mode will use the Strip Chart Duration, and Strip Chart Hours parameters to specify what duration of data to display.

If Disabled then History mode will be selected by default and will load the last 24 hours of data by default unless the user as specified a particular Date Range within the last 24 hours.

Strip Chart Duration (Hours)

If Strip Chart Mode is Enabled then this defines the number of hours to display back in time from the time of Refresh either manually or automatically.

If this parameter is set in the Trend Settings Dialog, then this value will be overridden for 24 hours in the browser session.

Left Axis Range (min, max)

If both min and max are set to 0.0 then auto range is used and the data retrieved will decide what the axis scaling is.

Min = bottom of axis scale

Max = top of axis scale

Right Axis Range (min, max)

If both min and max are set to 0.0 then auto range is used and the data retrieved will decide what the axis scaling is.

Min = bottom of axis scale

Max = top of axis scale

Boolean Top Margin (%)

Percentage of total graph height between boolean traces.

Boolean traces are scaled to the graph automatically based on percentage.

Trace Line Width

Controls how many pixels are used for all trace line thickness in the Trend.

Traces Shadow Depth

Number of pixels used for all trace shadows. A setting of 0 means no trace shadows.

Toolbar Background Color

Controls the background color of the Trend and Legend areas except inside the Trend.



If Toolbar Background Color, Gradient Start Color, and Gradient End Color are set to black (rgb:0,0,0) then the color is derived from the selected Page Theme.

Gradient Start Color

Controls the gradient start color (top) of the area inside the Trend.



If Toolbar Background Color, Gradient Start Color, and Gradient End Color are set to black (rgb:0,0,0) then the color is derived from the selected Page Theme.

Gradient End Color

Controls the gradient end (bottom) color of the area inside the Trend.



If Toolbar Background Color, Gradient Start Color, and Gradient End Color are set to black (rgb:0,0,0) then the color is derived from the selected Page Theme.

Averaging Decimal Places

Number of decimal digits after the decimal for the Average display numbers in the Stats Dialog.

Legend Display

Display Legend on Trend graph by default.

If this parameter is set in the Trend Settings Dialog, then this value will be overridden for 24 hours in the browser session.

Statistics Range Display

Show Minimum and Maximum values in the Stats Dialog.

If this parameter is set in the Trend Settings Dialog, then this value will be overridden for 24 hours in the browser session.

Statistics Average Display

Show Trace Average calculation in the Stats Dialog.

If this parameter is set in the Trend Settings Dialog, then this value will be overridden for 24 hours in the browser session.

Cursor Hover Display

Show mouse hover value from cursor position in Legend display.

If this parameter is set in the Trend Settings Dialog, then this value will be overridden for 24 hours in the browser session.

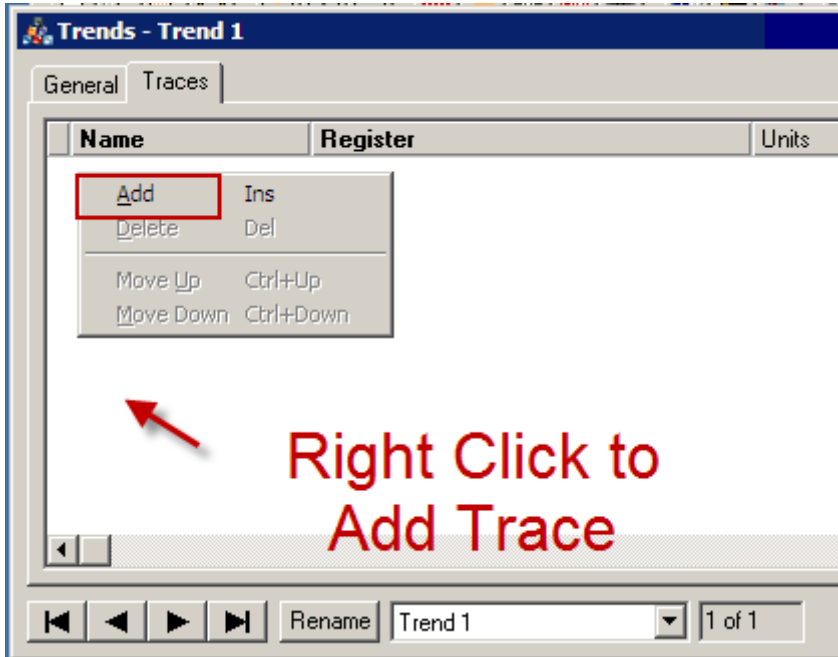


The Legend display must be enabled for this to work.

Configuring Traces and Trace Parameters

You can add up to 64 traces for all trends at this time with the default web page. If you have more than 8 or 10, it is recommended that you use more than one Trend graph.

To Add a Trace, right click in the white grid area and click Add on the speed menu.

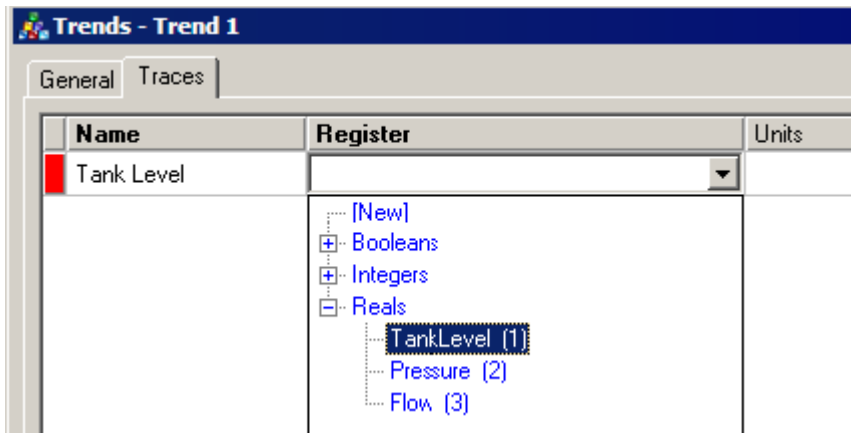


Or Click on the Add Entry button to the right.

Give your trace a name.

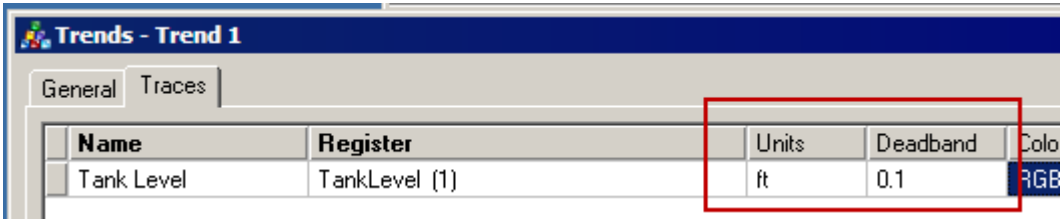


Select a register.

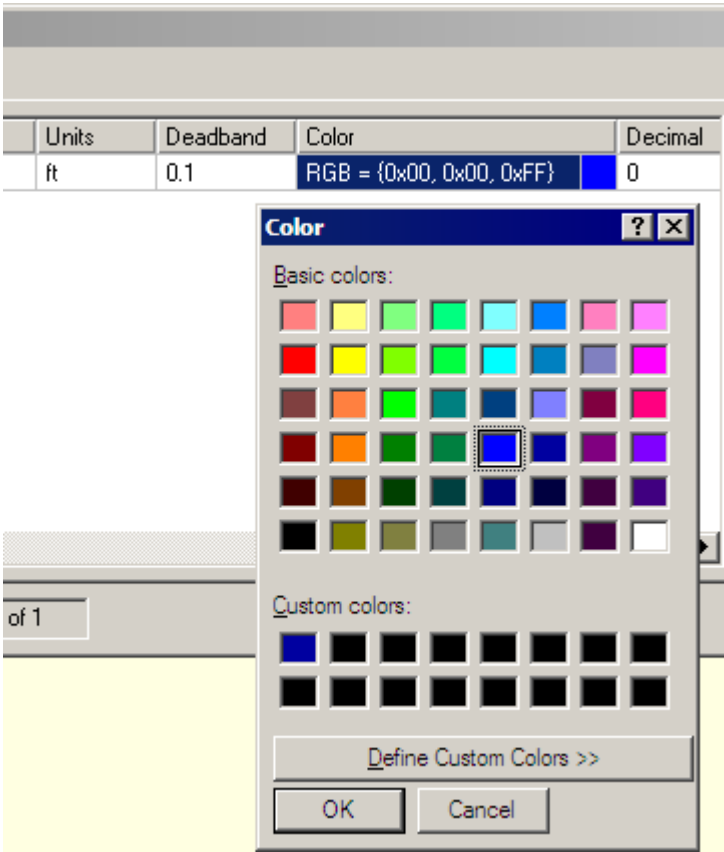


Enter Units if applicable (boolean values don't usually have units), you can leave it blank. These units show up in the Legend and Stats Dialog if configured.

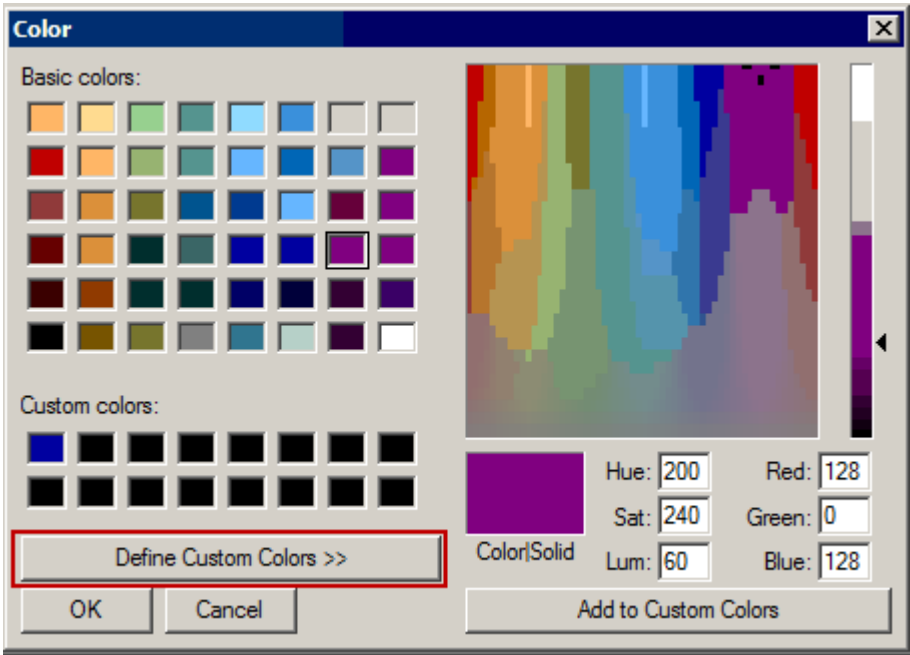
Enter a dead band if you only want to see a certain amount of change. For integers, the minimum dead band is one.



Select a color:



You can select a custom color as well:



If you are using a real number for your trace register, you can select the number of digits after the decimal place. If using an Integer number, the decimal place is displayed dividing the number by 10 per decimal place.

	Decimal	Height %
FF}	2	--

If you are using a boolean as your data source, you must select a height %. This is the percentage of the graph height you will use when going from false to true. The total of the Height %'s of all boolean traces + Height Top Margin * number of traces should not exceed 100%.

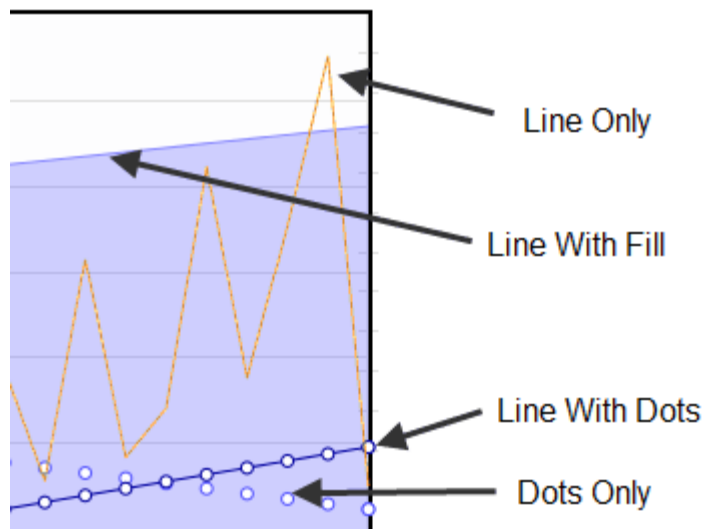
Select the axis, left or right that you want to apply the trace too.



Choose a line style that fits your needs, you can choose from the options below:

Line	Dot Rac	Fill	Bars
Solid	1	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Solid	1	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Dots	1	<input type="checkbox"/>	<input type="checkbox"/>
Both	1	<input type="checkbox"/>	<input type="checkbox"/>

Solid line, Dotted line or Both Solid with Dotted Line. Here is some idea of what they look like:



You can fill with a transparent color or fill with bars. These configurations can be used to make the lines very recognizable on the trend graph even when displayed simultaneously.

Accessing a Trend Web Page

After downloading the Trend configuration with the application, the trend must be accessed via a web browser.

To access a trend use the following URL:

Note: don't forget the trailing "/" if you access it this way:

`http://<ip address>/Trends/<Trend_Name>/`

or

`http://<ip address>/Trends/<Trend_Name>/index.html`

Spaces in the <Trend_Name> are replaced by "_".

If your trend name is "Trend 1" then the Trend_Name designation will be "Trend_1" then your URL would be:

`http://<ip address>/Trends/Trend_1/`

or

`http://<ip address>/Trends/Trend_1/index.html.`

Trends can also be accessed from the User Portal web interface. See *User Portal (Web Interface)* (on page 423) for details on how to set this up.

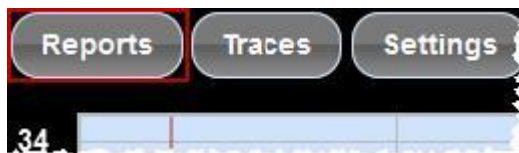
Reports Button

The Trending feature stores data to the local drive according to the sample rate setup in the Trend configuration. This data can also be access via the Reports button in the Trend Interface.

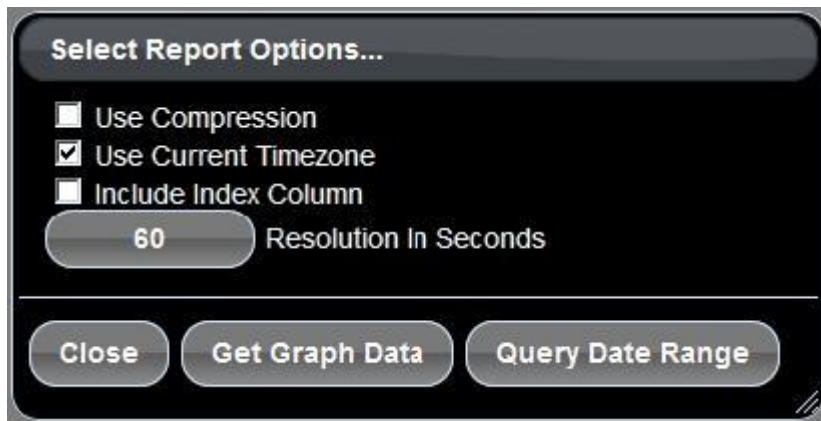
From the Reports dialog you can specify how you want the report formatted, then generate a .CSV file on the controller and download it to your PC.

Only Enabled traces are used in the report download. If you wish to exclude some Traces, disable them in the Traces dialog. See the **Traces Button** (on page 470) help for details.

To start, click on the Reports button in the Trend Interface.



This brings up the Report Options dialog.



Get Graph Data Button

Retrieves the data from the current graph by using the Trend's current configuration. If the Trend is in History mode and displaying a certain date range, then that is the data that will be retrieved in the report. The same is true for any Strip mode or Zoomed data.

Query Date Range Button

Brings up the Choose Date Range... dialog:

Choose Date Range...

Start Date: 08/21/2012 09:00 End Date: 08/22/2012 10:00

Enter date range of report.

Cancel Download

By Default, the date range will come up as the last 24 hours. Click on the Start Date or End Date date or hour to change them.

Choose Date Range...

Start Date: 08/21/2012 09:00 End Date: 08/22/2012 10:00

August 2012

Su Mo Tu We Th Fr Sa

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Hour

AM 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23

PM

Download

Once the date range is selected, click the Download button.

Use Compression

If you want to compress duplicate data when generating the report then only changed data will be loaded into the file.

Use Current Time Zone

If selected, the time zone of the browser is used for the date. If unchecked, GMT (UTC) is used instead.

Include Index Column

Includes an index column in the report to enumerate the records.

With Index Column:

Timestamp	Index	Tank Level (ft)	Clear Well (ft)	Pump 1 Run	Pump 2 Run
8/22/2012 14:12:00	0	30.1586	5.25102	1	0

8/22/2012 14:12:07	1	30.0764	5.25069	1	0
8/22/2012 14:12:14	2	29.9956	5.25046	1	0
8/22/2012 14:12:21	3	29.9149	5.25032	1	0

Without Index Column:

Timestamp	Tank Level (ft)	Clear Well (ft)	Pump 1 Run	Pump 2 Run
8/22/2012 14:12:00	30.1586	5.25102	1	0
8/22/2012 14:12:07	30.0764	5.25069	1	0
8/22/2012 14:12:14	29.9956	5.25046	1	0
8/22/2012 14:12:21	29.9149	5.25032	1	0

Resolution in Seconds

This is used when Query Date Range is selected and specifies how much resolution the downloaded report will have. The smaller the resolution, the more data that will be retrieved and the longer it will take the controller to extract and download.

Traces Button

Clicking on the Traces Button allows the user to select the traces they are currently viewing by disabling the ones configured in the Trend record in ScadaBuilder (see next section):



Settings Button

The Settings menu controls the Legend, Cursor, strip chart Refresh Time and the number of hours show in Strip mode:



The Min/Max and Average options

Controls the data that shows up in the Statistics dialog when you click on the Stats button.

Legend and Cursor Options

When the Legend and Cursor options are Enabled, you may hover over any point in the graph to see the data at the cursor position:



If the Legend is in your way, you can drag it off of the graph into an area of the web page that is unoccupied. The Legend will become opaque and stay at that position until the page is refreshed:



This functionality works for either the Left or the Right Legend.

The legend also reflects which Trace is applied to which Legend. Names on the Left Legend apply to the Left Axis Scale and the Right Legend Names apply to the Right Axis Scale.

Left And Right Minimum and Maximum Settings

These control the range of each Axis on the graph. If the Minimum and Maximum for an Axis are set to 0 then auto scaling is applied to that Axis. The Minimum and Maximum settings are an override to the ScadaBuilder Trend configurations and will remain at the set values for 24 hours, then will revert back to the ScadaBuilder configuration.

Refresh Time Seconds

When in Strip Mode, this controls how often the Trend updates. This setting is saved for 24 hours then reverts back to the ScadaBuilder Trend configuration.

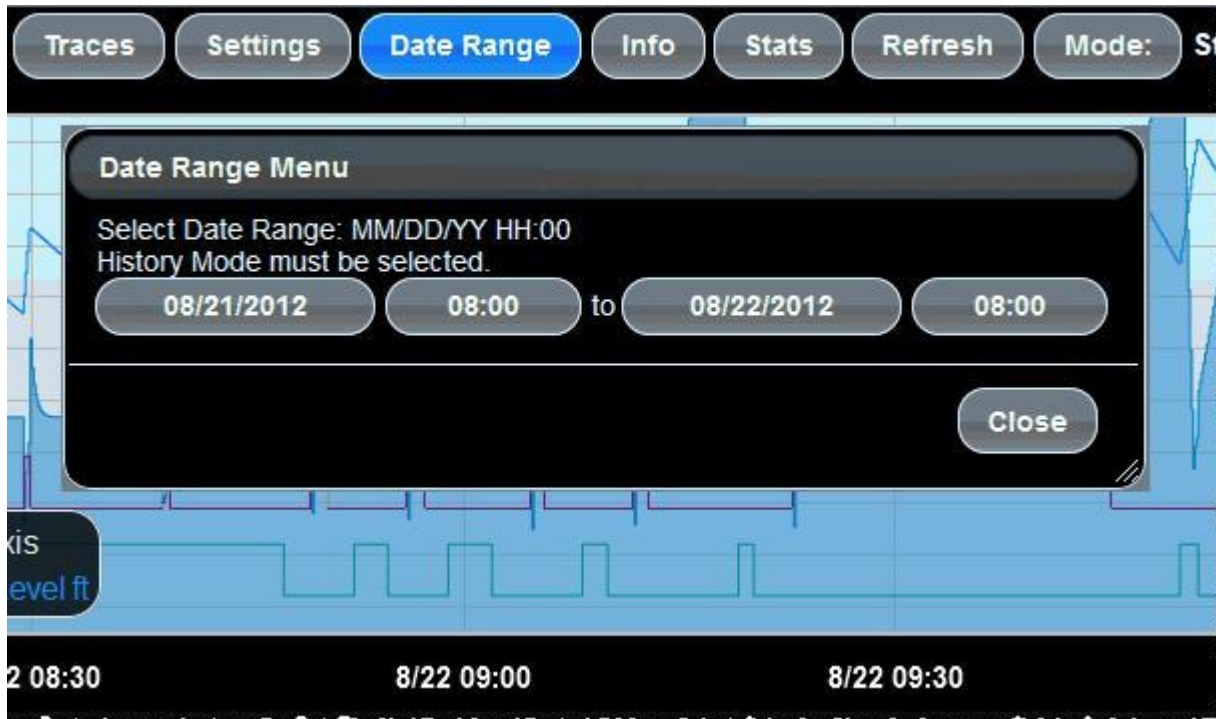
Refresh Display Hours

Controls how many hours of data are displayed in Strip Mode. This setting is saved for 24 hours then reverts back to the ScadaBuilder Trend configuration.

Date Range Button

Though you can select a date range at any time, to view the data in the range, you must have Historical Mode selected. See *Mode (Strip/History Selector)* (on page 478) section in the document.

Clicking on the Date Range button you are given the selection of two date/time combinations:



Clicking on the first field gives you a calendar to select the starting date of the trend display:



Clicking on any point in the box will change the date selected. The same is true for the hour fields:



The Start Date (first two fields) **MUST** be earlier than the End Date (second two fields). You are allowed to select them out of sequence if you like but the values will be qualified on the next Refresh in Historical Mode. Once you have selected the Start and End date/time, you must either select the Historical Mode from the Strip/Historical Mode: button. Clicking the Refresh on the web page will return the Trend to the default mode configured in ScadaBuilder.

Start Date, and End Date are stored in cookies in the browser for 24 hours from when the last Date Range was modified. After that the Date Range is set to 24 hours from the closest minute to the current time by default.

Info Button

After a successful get of the Trend data, the Info Button takes you to the Extended Information dialog which can give you details about the data in the trend and the status of the current data get:



Trace: Displays the Trace Name for each trace.

Points: Gives you a detail of how many points are used in each trace of the Trend.

Total Points: Shows the total number of time points displayed in the Trend.

Left and Right Axis Range: Date range of the actual data received. If in Historical Mode, not all of the Date Range may be used if there is no data to recall. The system will do its best to fill with what data it has stored on the controller's disk drive.

Load Time: Time in milliseconds it took for the last data retrieval from the controller.

Status: Current status of a get. If a data retrieval is in process, the command used will be displayed here.

Trend and Library software versions are also displayed.

Stats Button

The Stats Dialog will give you the latest value and the Min, Max, and Average of any currently enabled numerical traces. It also shows the latest value off each trace.



The start and end of the date range of the actual Trend data are also shown.

Refresh Button



The Refresh Button may be used in either Strip or History mode. It is disabled during a data retrieval operation but is particularly useful in History mode.

When zooming in on a date range from the graph, the Refresh button will reset the graph to the currently configured Date Range. See Date Range Button and Strip/History Button section for details.

Mode (Strip/History Selector)

The Strip and History Modes are mutually exclusive. The default mode is set in the ScadaBuilder Trend Dialog.

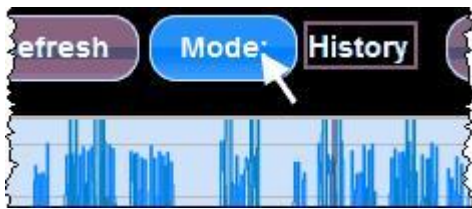
To Select Strip mode, simple click on the Mode: Button until the text next to the Mode: button shows "Strip".



If Strip mode is selected then the Refresh Time and Hours in the Settings Dialog control how often and how much Trend data is Refreshed. The Refresh Time says how often the data is retrieved. The Hours setting controls how far back in time to look for data.

When a Strip Mode retrieval is being done, the Mode: selector and Refresh button will be disabled.

To select History Mode, click on the Mode: Button until the text next to the Mode: button shows "History".

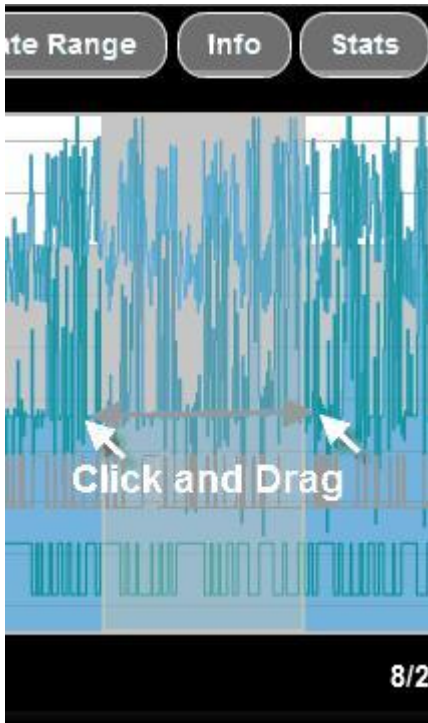


In History Mode, the Date Range (configured in the Date Range Dialog) is used to retrieve data. When the History Button is clicked, the Trend will automatically refresh to the Date Range configured. You will see a Loading... dialog. If you are using Firefox, a progress bar will show. In Internet Explorer, no progress bar is possible so you will simply see the Loading... status.

Zooming (PC Browsers)

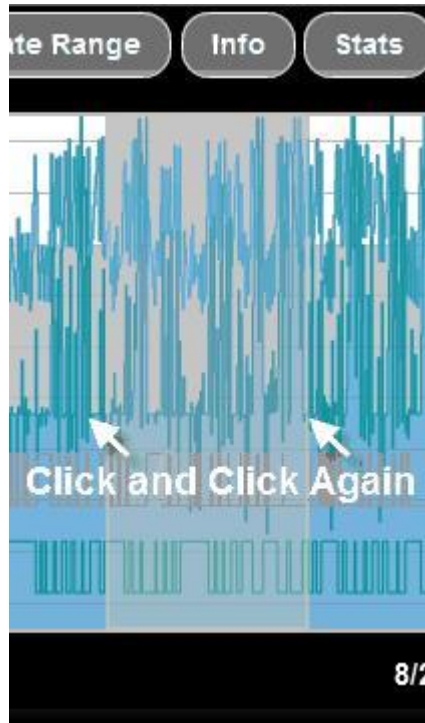
To Zoom within the Trend you must be in History Mode. Strip Mode does not support Zooming since the data could be refreshed at any time.

There are two modes to zooming controlled by the Zoom: button--Drag and Click



In Drag Mode you may use your mouse to click and drag a section of the Trend:

In Click Mode you may use your mouse to click and drag a section of the Trend.



The Selected area on the Trend is used as the new start and end time to get the data from the controller. You will see another "Loading..." status while the data is being retrieved. The Trend will then show the data selected.

You may Zoom as many times as you like to get more detailed data as you zoom in.

Clicking the Refresh Button will restore the original data selected in the Date Range Dialog.

Cursor: Mode Hovering and Zooming With iPad and iPhone Browsers

Since there is no "dragging" on the touch pad display for iPhone and iPad Safari browsers, the mechanism for Hovering and Zooming has to change a little.

In Strip Chart mode, clicking in the graph shows the cursor and updates the legends with the data and date informations.

In History Mode however, there are two Cursor: modes for interacting with the Trend Graph should the system detect that it is on an iPhone or iPad--Hover and Zoom.

When the text to the Cursor: button shows "Hover", clicking in the graph will act like a mouse over event and show the cursor at the clicked point on the graph and update the legends.

Clicking the Cursor: button will change the mode to "Zoom" in the text next to the button. The next two clicks in the graph will mark the zoom area. As soon as the second point is clicked, the graph will be reloaded with those two points as the designated date range.

The Cursor: mode is then changed back to "Hover".

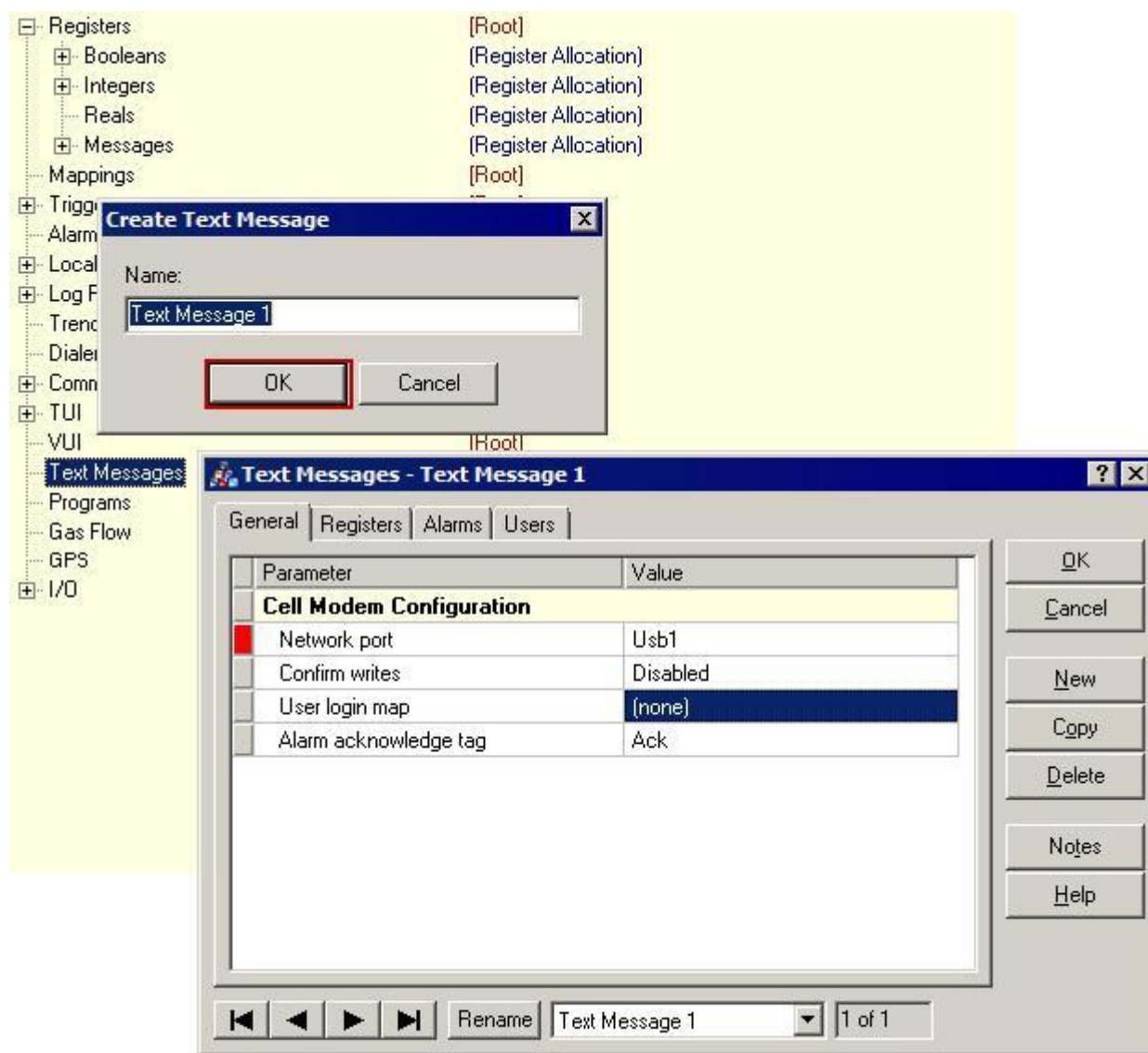
To zoom again, click the Cursor: button again to change the mode to "Zoom" and repeat the process.

The Refresh button will return the graph to the original (un-zoomed) date range.

Text Message Interface (TMI)

The Text Message Interface in conjunction with a Messenger Cell modem and Cell Modem Network Port can query a unit for register data and alarms as well as acknowledge alarms from any Text Message ready phone or interface. See *Using a Cellular Modem Network Port* (on page 210) section for details on configuring the Cell Modem Network Port.

To create a TMI, double click on the Text Message section in the ScadaBuilder Setup window or click the **Setup | Text Message Interface...** menu.



Select a Network Port with the Cell Modem connected.

For the moment, we will leave things at defaults just to get started.

Click on the Users Tab to setup a user account.



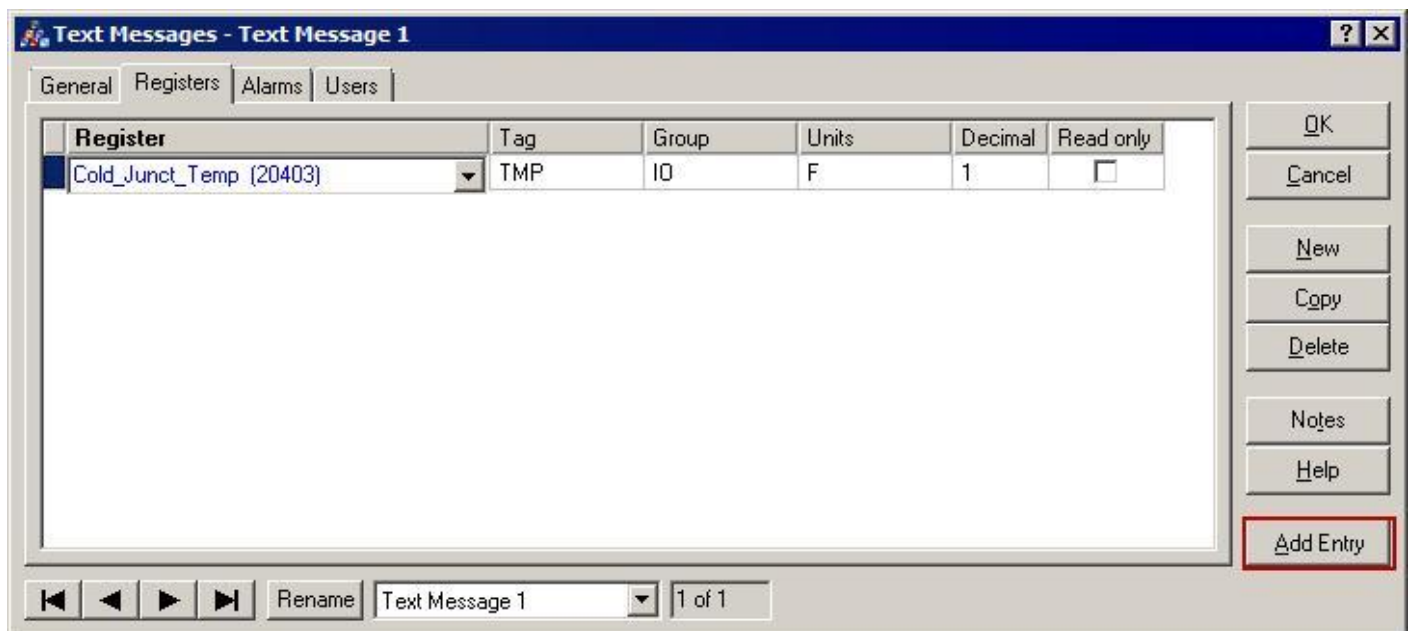
Click on the Add Entry button or right click and select Add.

Enter a Username and Password in the two fields.



Click on the Registers Tab to enter tags that may be accessed from the Text Message Interface (TMI).

Select a Register, Enter a Tag, Group, Units and number of Decimal places.



Download this to the unit and you are ready to access the TMI.

In This Section

Logging In To The TMI	485
Using More Complex TMI Grouping	485
Text Message Interface (TMI) Command Reference	487
Text Message Interface (TMI) Reference	490

Logging In To The TMI

First you must know the phone number of the cell modem you are talking to. This is programmed into the SIM card by your carrier or provider.

Text your user login like this.

Enter the Text Message Phone number.

In the message type ".admin.admin" without the quotes (your username and password will be different as configured in the previous section).

Send the message.

You will receive a response "Login:<Text Message Name>" to confirm that you are logged in.

Text your tag name or group name and send it. For this example we used "temp" as the tag name and "io" as the group name.

In this case the tag name comes back as "temp=<value>".

You have just successfully completed a basic Text Message Interface.

Using More Complex TMI Grouping

For our example Command Reference Section below, let's setup a more complex TMI configuration.

We will configure two Text Message Interfaces, one for normal user operation and the other for Joe the maintenance guy.

Open the Text Message Interface and configure the following registers and parameters:

General Registers Alarms Users						
Register	Tag	Group	Units	Decimal	Read only	
UIA1 (23)	UIA1	io	mA	3	<input checked="" type="checkbox"/>	
UIA2 (24)	UIA2	io	mA	3	<input checked="" type="checkbox"/>	
DI1 (20601)	DI1	io		0	<input checked="" type="checkbox"/>	
DI1 (20601)	DI2	io		0	<input checked="" type="checkbox"/>	
Tank1 (21)	Tank1	tanks	ft	1	<input checked="" type="checkbox"/>	
Tank2 (22)	Tank2	tanks	ft	2	<input checked="" type="checkbox"/>	
Pump1 (31)	Pump1	pumps		0	<input type="checkbox"/>	
Pump2 (32)	Pump2	pumps		0	<input type="checkbox"/>	
spTank2LowAlarm (35)	spT2LO	sp	ft	2	<input type="checkbox"/>	

Here we have three Groups: io, tanks, and pumps.

All are read-only except for the pumps and a low alarm setpoint for Tank1 which we want to control.

We also want to set up some alarms to go with our pumps. See *Using Alarms* (on page 159) for configuring and setting up a basic alarm system. But we want to access these alarms from the TMI. We go to the Alarms tab and setup this configuration.

General Registers Alarms Users			
Tag	Alarm state		
unack	Unacknowledged		
ack	Acknowledged		
idle	Idle		

With this configuration the "unack" tag will list out any Alarms that are Active.

The "ack tag will return all acknowledged Alarms.

The "idle" tag will list out all idle Alarms.

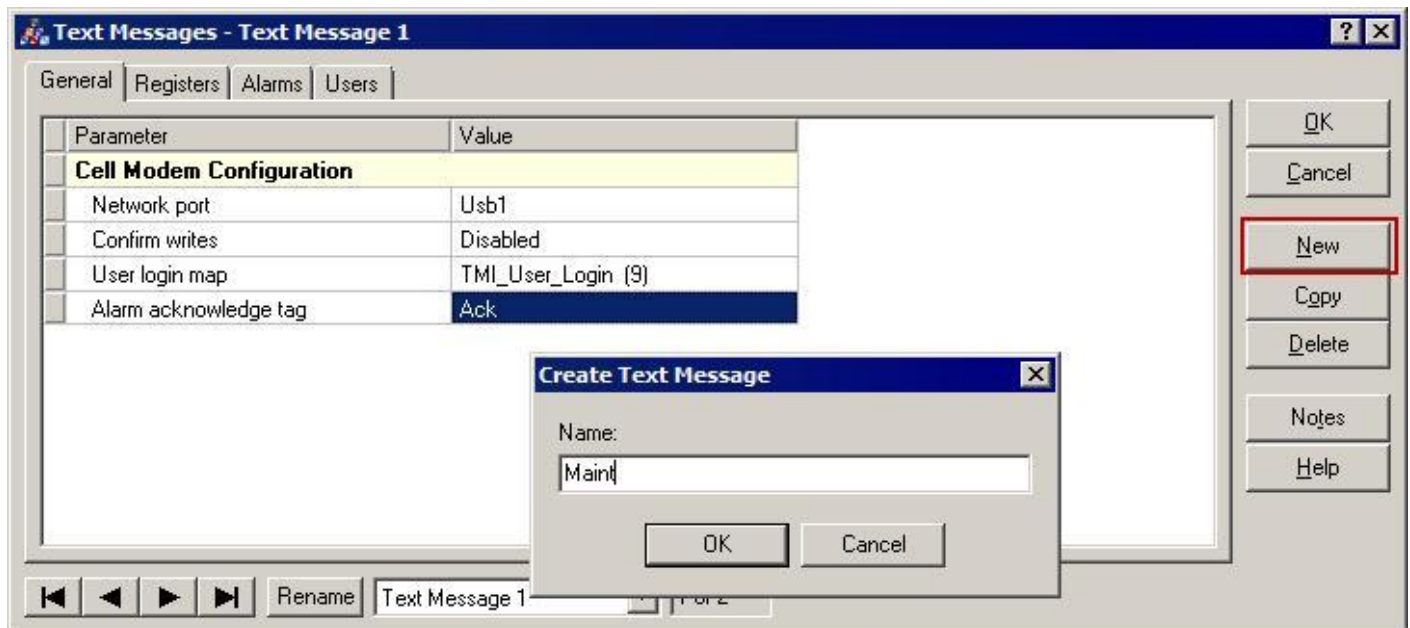
We move to the General tab and setup the acknowledge text code, in this case Ak (as we have already used ack in the tags above.

General | Registers | Alarms | Users |

Parameter	Value
Cell Modem Configuration	
Network port	Usb1
Confirm writes	Disabled
User login map	TMI User Login (9)
Alarm acknowledge tag	Ak

Setting Up A Second Maintenance Text Message Interface

Click on the New Button and setup a second Text Message Interface and name it Maint:



Select the same Cell Modem port as the first interface and click on the Users Tab:

General	Registers	Alarms	Users
Username		Password	
joe		je	

Here we put in Joe the maintenance guy's account.

Click on the Registers tab and we'll setup what Joe and only Joe is interested in seeing:

General Registers Alarms Users

Register	Tag	Group	Units	Decimal	Read only
Pres (25)	Pres	Group1	Psi	1	<input checked="" type="checkbox"/>
Pump1Cur (26)	Pump1Cur	Group1	Amps	1	<input checked="" type="checkbox"/>
Pump2Cur (27)	Pump2Cur	Group1	Amps	1	<input checked="" type="checkbox"/>

He has just three registers all on one group "Group1".

Once we download this, Joe can then log in to his own area to get his data and does not have to be bothered with the operator's requirements for data display.

Text Message Interface (TMI) Command Reference

This reference describes the Test Message Interface Syntax used to communicate from any phone capable of text messaging. It uses the TMI configurations in the previous section for both the commands and the responses

TMI Login Command

The login request supports many command configuration to aid in getting a job done in as few messages as possible. To enter the login command start with a "." followed by the user name, another "." and the password. Optionally you can

Request Format: .user.pass[.cfg][:oper1,oper2 .. ,operN]

Example Login Commands:	Description	Example Response
.admin.admin	Login only.	Login: Text Message 1
.admin.admin:Tanks	Login & read tank levels.	Login: Text Message 1 Tank1=32.8 ft Tank2=6.28 ft
.admin.admin:Tanks,Pumps	Login & read tank levels & pump states.	Login: Text Message 1 Tank1=32.8 ft Tank2=6.28 ft Pump1=0 Pump2=1
.admin.admin:Pump1=0,Pump2=1**	Login & turn pump 1 off & pump 2 on.	Login: Text Message 1 Pump1=0 Pump2=1
.admin.admin:unack	Login and get all unacknowledged alarms	Login: Text Message 1. Unacknowledged Alarms: Tank 2 Low Tank Alarm

Log In To a Specific Text Message Interface Configuration

..admin.admin.Text Message 1	Login into IO Text Message 1 config.	Login: IO UIA1=6.123 mA UIA2=15.682 mA DI1=1 DI2=0
.admin.admin.Text Message 1:UIA1	Login into IO Text Message 1 config & read UIA1 tag	Login: IO UIA1=6.123 mA
.admin.admin.Text Message 1:DO1=1**	Login into Text Message 1 Config & turn DO1 on.	Login: IO DO1=1

.joe.joe.Maint	Login into user's 2nd config Maint.	Login: Maint
.joe.joe.Maint:Group1	Login into user's 2nd config Maint and read Group1.	Login: Maint Pres=62.5 Psi Pump1Cur=42.1 Amps Pump2Cur=0.1 Amps

**Registers must be set to read/write (Read Only flag unchecked) to be able to write register values.

If Confirm is turned on, then the system will respond with a Confirm (Y/N)? Send N back to cancel and send Y back to confirm the write.

TMI Logout Command

Send a single period "." (no quotes) to logout of the TMI.

Response: Logged Out:

TMI Read Commands

Request Format: <tag1>[,<tag2> .. ,<tagN>]

Must be logged in to a configuration (Text Message Configuration) for read command to work.

Read tags can be listed out in a comma delimited format and include alarm tags.

Example Read Commands:	Description	Example Response
Tanks,Pumps	Read Tanks and Pumps groups.	Tank1=32.8 ft Tank2=6.28 ft Pump1=0 Pump2=1
IO	Read the IO Group	UIA1=6.123 mA UIA2=15.682 mA DI1=1 DI2=0
unack	Read All unacknowledged Alarms	Tank 2 Low Alarm
ack	Read All Acknowledged Alarms	Acknowledged Alarms: (none)
idle	Read All Idle Alarms	Idle Alarms: Pump 1 Fail Alarm Pump 2 Fail Alarm Tank1 Low Alarm

TMI Write Commands

Request: <tag1>=<val1>[,<tag1>=<val2> .. ,<tagN>=<valN>]

If Confirm is turned on, then the system will respond with a Confirm (Y/N)? Send N back to cancel and send Y back to confirm the write.

Example Write Commands:	Description	Example Response
Pump1=1 or Pump1#1**	Turn on Pump1	Pump1=1 Confirm (Y/N)?
Y	Confirm the write.	Confirmed.
spT2Lo=10.2	Write the Tank 2 Low Alarm Setpoint	spT2LO=10.2 Confirm (Y/N)?
Y	Confirm the write.	Confirmed.
Ak=All	Acknowledge All Alarms	Confirm (Y/N)?
Y	Confirm the write.	Confirmed.
Ak=Alarm1,Ak=Alarm2	Acknowledge Alarm 1 and Alarm 2	Confirm (Y/N)?
Y	Confirm the write.	Confirmed.

Text Message Interface (TMI) Reference

General Tab

Network Port

Network Port can be configured as a serial or USB port. Designed to connect to a Messenger Series Cell Modem with a Text Message Enabled account configured in the SIM card. See *Using a Cellular Modem Network Port* (on page 210) for details on setting up a Cellular Network Port.

Confirm Writes

Use the to enable write confirmation on any Text Message Interface write commands or alarm acknowledge commands.

User Login Map

Map a Message register to display what account is currently logged in to the TMI. The Message register will display the <user account>:<text message interface name> in the string.

Alarm Acknowledge Tag

This is the string code to enter an alarm acknowledge command. See *Text Message Interface (TMI) Command Reference* (on page 487) for more details. Typical format if the string is set to "ack" is:

```
ack=all
ack=alarm1
ack=alarm1,ack=alarm2
```

If confirmation is turned on then the answer will be Confirm (Y/N)?

Send a Y to confirm the alarm acknowledge.

Registers Tab

Register Column

Select a Register from the database for each TMI tag.

Tag Column

Text tag to access values in the selected Register from the TMI.

Group Column

Configure the tag for a group. All tags with a matching group may be read from the TMI in one command.

Units Column

String which is displayed at the end of a register value after it has been read back from the TMI.

Example:

If the units is "mA" then a tag will look like:

```
UIA1=4.000 mA
```

Decimal Column

If a tag is configured from a Real (floating point) register, this configures how many decimal places are shown when the tag is read.

Read-Only Checkbox Column

Check this box if you want the tag to be read-only. All writes to the tag will be denied.

Alarms Tab

Alarm State Tags

Alarm state tags represent the text enter when retrieving alarm status. Any tag may be used to get any group of alarm states.

The defaults are shown here. Any text may be used for each tag.

General Registers Alarms Users			
Tag	Alarm state		
unack	Unacknowledged		
ack	Acknowledged		
idle	Idle		

Users Tab

Username

User name that will be used to log in to this Text Message Interface. You can have as many users as you like, each with their own password.

Password

Password for the associated Username on this line.

Mappings Reference

Mappings include hardware specific functions that apply to the particular controller specified in the Node Settings - General tab.

Node Mappings

Local time:

Time zone map:
TimeZone (5001)

Daylight savings map:
DaylightSavings (5003)

Register-to-device mappings:

Reboot map:
Reboot (1227)

Memory available map:
MemoryAvailable (1304)

Status LED map:
(none)

Clock speed set/map:
(high speed)

Device-to-register mappings:

RTC serial number map (LSBs):
(none)

RTC serial number map (MSBs):
(none)

Low battery map:
(none)

Controller serial number map:
(none)

OK

Cancel

Help

In This Section

Reboot Map	493
Status LED Map.....	494
Memory Available Map.....	494
Clock Speed Set / Map	494
Device To Register Mapping	494
Time Zone Map/Constant	495
Daylight Savings Map	495
Controller Serial Number Map.....	495
FTP Server Status Buffer	496
FTP Client Status Buffer	496
HTTP Server Status Button.....	496

Reboot Map

Maps a Boolean register to the system reboot control. Set the selected register to value of 1 (TRUE) to reboot the system at runtime.

Status LED Map

Maps a Boolean register to the status LED. Set the selected register to value of 1 (TRUE) to turn the LED on, and a value of 0 (FALSE) to turn it off.

Memory Available Map

Maps an integer register that gets loaded with the amount of available memory in the system when the controller starts up. This can be used as an indicator by the user to determine when an application is getting close to running out of memory during application development.



It is recommended to leave about 10,000 bytes of memory for dynamic allocations after the application has started.

Clock Speed Set / Map

This item allows the configuration of the processor clock speed for controllers that support it. The settings are:

- | | |
|-------------------------|---|
| <i>High Speed</i> | This offers the best processing performance but also the most power consumption. For process intensive operations where low power operation is not a concern this is the best option. |
| <i>Low Power</i> | This offers the lowest power consumption but the worst processing performance. It should be used where processing time is not a concern but power is limited such as solar and battery operation. |
| <i>Boolean Register</i> | This option offers the ability to change the processor speed on the fly. If the system has suddenly lost power and has to go to battery backup, setting the boolean to FALSE, the controller will run at the default highest possible clock speed. Setting the boolean to TRUE, the controller will run a lowest possible clock rate to reduce power. |

Device To Register Mapping

This feature allows direct access to some of the hardware registers available in the system.

Low Battery Map

This is a boolean value that can be mapped to a register and can determine the health of the RTC battery which is also responsible for battery backed ram where non-volatile registers are stored. A FALSE means the battery is okay while a TRUE means the battery is going low and should be replaced. Some controllers support changing the battery (coin type battery) and others do not. Contact ICL technical support should you have any questions about your particular model.

Serial Numbers

Each controller has a build in serial number that identifies it as an individual from all other controllers. The number is a 64 bit number and must be read in as two integer registers. If the numbers are represented as hexadecimal bytes (two characters 0-F) then the following is the representation of what is placed in each register:

Serial Number Map LSB's

xx xx xx xx xx xx xx xx

Serial Number Map MSB's

xx xx xx xx xx xx xx xx

This feature may be used to protect a program from being transferred and copied illegally to another controller of like model.

Time Zone Map/Constant

The "Time Zone Map" control links an analog register (integer or float) that can be used to specify the local time zone that applies to the controller. This allows the user to change the time zone programmatically at run time.

If "(constant)" is selected from the "Time Zone Map" list then time zone can be selected from one of the choices on the "Time Zone Constant" list. Using a constant time zone will not allow the value to be changed at runtime.

The time zone is necessary to globally synchronize the system's date/time for such tasks such as sending Emails accessing the GPS, and recording time/date stamps in log files.

Daylight Savings Map

The "Daylight Savings Map" control links a Boolean register that can be used to specify the daylight savings mode that applies to the controller. This allows the user to change the mode programmatically at run time. Setting the register to TRUE indicates that daylight savings should be applied (adding 1 hour to GMT or whatever time zone is configured.).

Daylight savings is necessary to globally synchronize the system's date/time for such tasks such as sending Emails accessing the GPS, and recording time/date stamps in log files.

Controller Serial Number Map

The Controller Serial Number Map control allows the user to assign an integer register to expose the internal serial number of the controller. This serial number is a 6 digits or more and located on the controller's main processor board. ICL's customer service can reference this serial number in a variety of ways. They can lookup the order shipped, calibration data and shipping date for example. They can also lookup any type of RMA history as well.

FTP Server Status Buffer

The FTP server status buffer can be used to select a buffer register (message) that holds the current status of the FTP server connection. Connection, directory and file transfer information will be loaded into the buffer as appropriate. This is primarily a debugging aid that is typically displayed on a TUI screen.

FTP Client Status Buffer

The FTP client status buffer can be used to select a buffer register (message) that holds the current status of the FTP client connection. Login and file transfer information will be loaded into the buffer as appropriate. This is primarily a debugging aid that is typically displayed on a TUI screen.

HTTP Server Status Buffer

The HTTP server status buffer can be used to select a buffer register (message) that holds the current status of the HTTP server connection. Status information will be loaded into the buffer as appropriate. This is primarily a debugging aid that is typically displayed on a TUI screen.

Global Positioning Satellite (GPS) Interface

ICL Controllers support a direct interface with GPS (Global Positioning Satellite) receivers that support the NMEA-0183 standard (most GPS manufacturers support this standard). The GPS interface allows information from a GPS receiver to be “mapped” to registers in the controller. The GPS information includes date and time (UTC), latitude, longitude and altitude, as well as indicators to detect the validity of the incoming data.

To create a GPS interface, double-click on “GPS” in the Project Manger. A window will pop up for configuring the GPS interface.

Configure:

Network port:
Com3

RTC sync period:
0 (min)

Task period:
1000 (ms)

Comm error register:
GPSCommError (201)

Comm timeout:
30 (sec)

Data mapping:

Time buffer:
TimeBuffer (501)

Date buffer:
DateBuffer (502)

Latitude register:
Latitude (602)

Longitude register:
Longitude (601)

Altitude register:
Altitude (603)

Quality indicator register:
Quality (604)

Satellite count register:
SatCountReg (607)

Horizontal dilution reg:
HorizontalDilution (605)

Geoidal separation reg:
GeoSepReg (608)

Delta mapping:

Delta latch (Boolean):
DeltaLatch (701)

Delta time (ms):
DeltaTime (609)

Delta distance (feet):
DeltaDistance (610)

Delta threshold:
0.0 (degrees)

Supported GPS messages:

GGA - GPS Fix Data (time, long, latd, alt, qual, sat, dil, sep)
RMC - GPS Transit Data (time, date, long, latd)
ZDA - Time & Date (time, date)

In This Section

GPS Reference..... 497

GPS Reference

The GPS (global positioning satellite) interface allows information from a GPS receiver to be mapped to specified registers in the controller. The GPS information includes fields such as date/time (UTC), latitude, longitude and altitude. The GPS also provides indicators to detect the validity of the incoming data.

NOTE: The GPS must be able to support the NMEA- 0183 protocol with the message subset as listed in the GPS dialog box.

Network Port

Identifies which Network Port to use for the GPS interface. The standard port setting for an NMEA 0183 communication interface are 4800 baud, 8 data bits, 1 stop and no parity.

RTC Sync Period

The RTC synchronization period specifies the rate in minutes at which the real time clock is updated from the GPS data. The date and time output from the GPS is known as UTC (universal time code) which is equivalent to GMT (Greenwich Mean Time). When the RTC is updated the date and time will be adjusted for the local time based on time zone and daylight savings settings configured in the Node Settings dialog.

Setting the value to '0' will disabled synchronization of the RTC from the GPS.

Comm Error Register

The comm error register allow a Boolean register to be mapped that indicates when a GPS communications error has occurred. If a supported message has not been received from the GPS unit within the duration specified by the comm timeout control value, the register will be set to TRUE indicating a comm error. When the next valid message is received from the GPS the register will be cleared back to FALSE.

Comm Timeout

The comm timeout value specifies the time in seconds to wait before indicating a communication error with the GPS receiver. If a message is not received from the GPS in the specified time then the comm error register will be set to TRUE. If a comm error register is not selected then this control will be disabled.

Task Period

How often to run the GPS task. The greater the resolution of delta positioning needs to be, the lower this time should be.

Data Mapping

The data mapping controls allows information from the GPS to be loaded into registers that can be accessed by the control program. The data that can be mapped is described as follows:

<i>Time Buffer</i>	The UTC (Universal time code) can be mapped to a buffer in the format "hh:mm:ss.ss".
<i>Date Buffer</i>	The date can be mapped to a buffer in the format "mm-dd-yyyy".
<i>Latitude Register</i>	The latitude can be mapped to a register in the format "dddmm.mmm" (where "ddd" is degrees and "mm.mmm" is minutes and fraction of minutes). North is positive and South is negative.
<i>Longitude Register</i>	The longitude can be mapped to a register in the format "dddmm.mmm" (where "ddd" is degrees and "mm.mmm" is minutes and fraction of minutes). East is positive and West is negative.
<i>Altitude Register</i>	The altitude can be mapped to a register in the format "x.x" Meters.

<i>Quality Indicator Register</i>	The quality indicator can be mapped to a register with the following values: 0=No GPS, 1=GPS, 2=DGPS. It takes at least three satellites to triangulate position. The quality indicator will output a value of 0 until the satellite count is at least three.
<i>Satellite Count Register</i>	The satellite count can be mapped to a register. This indicates the number of satellites in use by the GPS. It takes at least three satellites to triangulate position. When three or more satellites are in use the quality indicator will output a non-zero value.
<i>Horizontal Dilution Register</i>	The horizontal dilution can be mapped to a register in the format "x.x". This is the horizontal dilution of precision (HDOP).
<i>Geoidal Separation Register</i>	The geoidal separation can be mapped to a register in the format "x.x" Meters. This is the difference between the WGS-84 earth ellipsoid and mean-sea-level.

Delta Mapping

<i>Delta Latch</i>	When on, the boolean will suspend delta sampling from GPS.
<i>Delta Time</i>	Register to set millisecond sampling rate for GPS delta position.
<i>Delta Distance</i>	Register to place result of GPS change in position (delta).
<i>Delta Threshold</i>	Adjusts the sensitivity of the delta change detection.

Required GPS Messages

This box lists the required NMEA 0183 messages that the GPS receiver must output to support the desired data that has been mapped to registers. The "RTC sync period" control will also affect the messages listed.

It is up to the user to configure his specific GPS receiver to output the listed messages. It is ok to output additional messages from GPS as the controller will simply ignore them. Though, additional messages will slightly slow throughput as the unused messages must still be parsed by the controller.

ISaGRAF Function and Function Blocks

As part of the ScadaWorks package, various supplemental functions and function blocks are provided that can be called from your ISaGRAF Programs. These functions and function blocks are described in this chapter. Rather than continuously referring to "C Functions" and "C Function Blocks", they will both be called "functions" in this chapter. Distinctions will be made between the two forms in the function definition table (see next section).

The functions provided may be called from within any ISaGRAF language.

For more information on ISaGRAF programming, please refer to the ISaGRAF Workbench Manual 3.40

http://www.iclinks.com/public_ftp/DocRelease/ISaGRAFWorkbench/v3.40/ISaGRAFWorkbench3.40.pdf

For Pinnacle and later controllers, please refer to the ISaGRAF 5.20 Manual

http://www.iclinks.com/public_ftp/DocRelease/ISaGRAFWorkbench/v5.20/WorkbenchV5.20.pdf

In This Section

ISaGRAF Data Types and Function Prototypes	501
Low Level Communications Functions	503
File I/O Functions	526
Network Control Functions.....	553
File Transfer Protocol (FTP) Functions	564
Low Level I/O Port Access Functions	568
Bit Packing and Unpacking functions.....	571
Instrumentation Functions	574
Variable Access Functions.....	590
Logical Functions.....	594
Logging Functions	595
Real Time Clock (RTC) Functions.....	599
Redundancy Function Block For Legacy Controllers	606
Redundancy Function Block for Pinnacle Controllers	612
ISaGRAF Socket Functions (UDP and TCP/IP).....	628

ISaGRAF Data Types and Function Prototypes

ISaGRAF presents data types with a slightly different nomenclature than ICL software. They are defined here:

For ISaGRAF 3 controllers (Etherlogic, ScadaFlexPlus and ICL 4300)

Data Type Name	Usage	Definition
BOO	TRUE or FALSE	boolean value
ANA	Numbers with no decimal point, i.e. 100 or 638.	32-bit Integer value
REAL	Numbers with a decimal point, i.e. 100.32 or 2365.62	32-bit Floating point value
TIMER	time representation up to 24 hours i.e. t#12h15m or t#60s	32-bit Timer values (not accessible from ScadaBuilder)
MESSAGE	strings, E-mail addresses, names, phone numbers etc. 'Steve' or '1 800 429-8234'	buffer, message or string.

For ISaGRAF 5 Controllers (Pinnacle and later)

Data Type Name	Usage	Definition
BOOL	TRUE or FALSE	boolean value
DINT	Numbers with no decimal point, i.e. 100 or 638.	32-bit Integer value
REAL	Numbers with a decimal point, i.e. 100.32 or 2365.62	32-bit Floating point value
TIMER	time representation up to 24 hours i.e. t#12h15m or t#60s	32-bit Timer values (not accessible from ScadaBuilder)
MESSAGE	strings, E-mail addresses, names, phone numbers etc. 'Steve' or '1 800 429-8234'	buffer, message or string.

The data type name is used to define function prototypes for passing data to functions and function blocks as well as returning data from them. The structure of the prototype is as follows:

<Return Value Type> <Function Name>(<Argument 1>, <Argument 2>, <Argument ...>);

For example, the following function allows the user to read how many bytes there are in a currently open Comport's software buffer

ANA ComRCnt(ANA port);

The ANA calls out the data type both passed and returned from the function. This means that the "port" parameter is an integer and you must place an integer register or constant as the port number and it returns an integer value that is stored into an integer register.



If using function on a Pinnacle or later series, all "ANA" data types have been converted to "DINT". Just substitute the ANA definition for a DINT definition. The function help will note if it is no longer supported on Pinnacle or later controllers.

All ICL functions are defined this way.

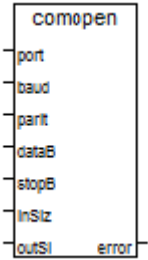
Function Definition Key

Type	C Function or C Function Block (this will tell you in which list in ISaGRAF to look for this function or function block).
Function	This is a definition of what the function does,
FBD	This will show the graphical depiction of what the block looks like in FBD code.
Syntax	This will show the prototype of function showing names and data types of the arguments and return parameters. If it is a function block, then the return value types are shown here.
Parameters	Definition of each parameter and its acceptable range of input.
Return Values	The return value or values and its or their types.
Remarks	Special considerations and cautions for using the function or function block.
ST Example	Structured text example of code for the function or function block. (* Code comments are shown like this and may be pasted directly in your program *)

Low Level Communications Functions

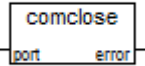
Low level communications functions are provided to allow you to perform your own serial communications independent of the protocol drivers that are built into the ISaGRAF Runtime Kernel

Open a Communications Port -- ComOpen()

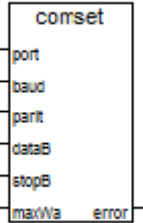
Type	C Function
Function	Opens an ICL-4300 serial port for reading/writing.
FBD	
Prototype	ANA ComOpen(ANA port, ANA baud, ANA parity, ANA dataBits, ANA stopBits, ANA inSize, ANA outSize);
Parameters	<p>port: COM port to open 1-7</p> <p>baud: baud rate to use 300,600,1200,2400,4800,9600,19200,38400,57600,115200</p> <p>parity: parity to use 0 = none 1 = odd 2 = even 3 = mark 4 = space 5 = "9-bit" mode, mark identifies first byte (address byte) 6 = "9-bit" mode space identifies first byte (address byte)</p> <p>dataBits: data bits to use 7, 8</p> <p>stopBits: stop bits to use 1, 2</p> <p>inSize: size of input (receive) buffer 32-65535</p> <p>outSize: size of output (transmit) buffer 32-65535</p>
Return Value	error: 0 = success, nonzero = failure

Remarks	<p>ComOpen must be called first before a serial port can be read or written using the other Com functions.</p> <p>The so-called "9-bit" mode (parity = 5, or parity = 6) is for use only with specialized equipment that requires it. In this mode, the first byte of each message is interpreted as an Address byte, and is identified as such by forcing the parity bit to either mark or space. The remaining bytes of the message (Data bytes) have the parity bit forced to the opposite state. The "polarity" of the parity values used is determined by the parity parameter to this function. If parity is set to 5, the address byte is identified with mark (1) parity. If parity is set to 6, the address byte is identified with space (0) parity. Special support for "9-bit" mode is only implemented for transmitting, and not for receiving (incoming bytes are not discarded based on parity). To properly transmit messages in this mode, you must allow the transmission of one message to complete, before you begin the transmission of another. If you do not, the parity bit will not be set properly for the first byte of the second message. The function ComXmtEm can be used to determine if a transmission is complete.</p>
ST Example	<pre>(* open COM2 with this configuration: *) (* 9600 baud, no parity, 8 data bits, 1 stop bit, input / output buffer size of 512 bytes: *) result := ComOpen(2, 9600, 0, 8, 1, 512, 512);</pre>

Close a Communications Port -- ComClose()

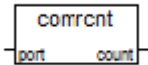
Type	C Function
Function	Close a serial communications port that was previously opened using ComOpen.
FBD	
Syntax	ANA ComClose(ANA port);
Parameters	port: ANA which COM port to close 1-7
Return Value	error: ANA 0 = success, nonzero = failure
Remarks	This function may be called when a process is done using the serial port.
ST Example	<pre>(* close COM3: *) result := ComClose(3);</pre>

Reset Communication Port Parameters -- ComSet()

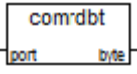
Type	C Function
Function	Change the settings associated with a serial port.
FBD	
Syntax	ANA ComSet(ANA port, ANA baud, ANA parity, ANA dataBits, ANA stopBits, ANA maxWait);
Parameters	<p>port: COM port to set 1-7</p> <p>baud: baud rate to use 300,600,1200,2400,4800,9600,19200,38400,57600,115200</p> <p>parity: parity to use 0 = none 1 = odd 2 = even 3 = mark 4 = space 5 = "9-bit" mode, mark identifies first byte (address byte) 6 = "9-bit" mode space identifies first byte (address byte)</p> <p>dataBits: data bits to use 7, 8</p> <p>stopBits: stop bits to use 1, 2</p> <p>maxWait: Maximum time to wait (milliseconds, -32768 to 32767) for any transmission to complete before changing settings. A negative number means wait indefinitely. 0 means do not wait.</p>

Return Value	error: ANA 0 = success, nonzero = failure
Remarks	This function allows you to change the settings of a serial port "on the fly." This function will wait for any current transmission to complete before the settings are changed.
ST Example	<pre>(* Open the port *) result := ComOpen(2, 9600, 0, 8, 1, 512, 512); . . . (* Increase baud rate later, waiting up to 100ms *) result := ComSet(2, 38400, 0, 8, 1, 100);</pre>

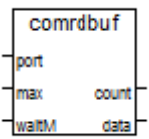
Get The Number of Bytes In The Receive Buffer -- ComRCnt()

Type	C Function
FBD	
Function	Checks an ICL-4300 serial port for any received bytes. Does not remove any bytes from the input buffer.
Syntax	ANA ComRCnt(ANA port);
Parameters	port: which COM port to check 1-7
Return Value	count: Returns the number of bytes that are in the receive buffer.
Remarks	Use ComRdBt, ComRdBts or ComRdBuf to actually read the byte(s) from the port.
ST Example	<pre>(* get a byte from COM5 if one has been received: *) if ComRCnt(5) > 0 then byte := ComRdBt(5); end_if;</pre>

Read A Byte Out Of The Receive Buffer -- ComRdBt()

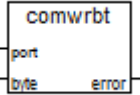
Type	C Function
Function	Read a byte from an ICL-4300 serial port.
FBD	
Syntax	ANA ComRdBt(ANA port);
Parameters	port: ANA which COM port to read from 1-7
Return Value	Byte: The byte which was read.
Remarks	If no byte is available, this function will return a null (\$00) value. In some cases, \$00 may be a valid value. If this is the case, you must first call ComRCnt to see if there is really a byte available.
ST Example	(* read a byte from COM4: *) byte := ComRdBt(4);

Read Multiple Bytes Out Of The Receive Buffer -- ComRdBuf()

Type	C Function Block
Function	Read multiple bytes from an ICL serial port.
FBD	
Syntax	ANA ComRdBuf(ANA port, ANA max, ANA waitms); returns: ANA ComRdBuf.count; MSG ComRdBuf.data;
Parameters	port: which COM port to read from 1-7 max: maximum number of bytes to read 0-255 waitMs: message variable to accept the bytes which are read -1 to 32767 maximum time to wait for max bytes
Return Values	count: ANA number of bytes read from the buffer. data: MSG message string with data read from the buffer.

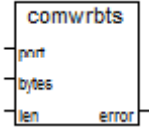
<i>Remarks</i>	<p>This function will return immediately with as many bytes as are available in the serial port receive buffer, up to the maximum specified. The bytes are stored to the message variable that you pass as the 2nd parameter.</p> <p>If you specify a value for max that is less than 1 or greater than 255, no bytes will be returned.</p>
<i>Cautions</i>	To use the function block, you must declare a function block instance for each operation used.
<i>ST Example</i>	<pre>(* ComRdStr is a function block instance of ComRdBuf() *) (* read up to 10 bytes from COM5 with no waiting: *) result:= ComRdStr(5, 10, 0); (* get the number of bytes that have been read out *) ByteCount := ComRdStr.count; (* store the data to a message variable *) MessageData := ComRdStr.data;</pre>

Write A Byte To The Transmit Buffer -- ComWrBt()

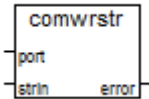
<i>Type</i>	C Function
<i>Function</i>	Writes a single byte to a serial port.
<i>FBD</i>	
<i>Syntax</i>	ANA ComWrBt(ANA port, ANA byte);
<i>Parameters</i>	<p>port: which COM port to write to 1-7</p> <p>byte: byte to write</p>
<i>Return Value</i>	error: ANA 0 = success, nonzero = failure
<i>Remarks</i>	Before using this function, you must open the port with ComOpen. If you have a sequence of bytes or a sequence of ASCII characters to send, consider using the ComWrBts or ComWrStr function, respectively. If a value greater than 255 is passed, only the least significant byte will be used.
<i>ST Example</i>	<pre>(* Send a hex A9 byte out COM2: *) result := ComWrBt(2, 16#A9);</pre>

Write Multiple Bytes To The Transmit Buffer -- ComWrBts()

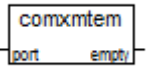
<i>Type</i>	C Function
-------------	------------

<i>Function</i>	Writes a sequence of bytes to an ICL-4300 serial port.
<i>Syntax</i>	ANA ComWrBts(ANA port, MSG bytes, ANA len);
<i>FBD</i>	
<i>Parameters</i>	<p>port: which COM port to write to 1-7</p> <p>bytes: bytes to write</p> <p>len: length of byte sequence</p>
<i>Return Value</i>	error: ANA 0 = success, nonzero = failure
<i>Remarks</i>	Before using this function, you must open the port with ComOpen. If you only have ASCII characters to send, consider using the ComWrStr function, which doesn't require a length parameter. However, ComWrBts must be used for messages with embedded NULL (\$00) characters. If len is out of range, nothing will be transmitted, and the function will return failure.
<i>ST Example</i>	<pre> (* Send a 16-bit value, represented in binary, embedded * in a message that begins with a NULL character (\$00) * and ends with a \$FF character. *) intVal := 1234; (* number to send *) (* Get the bytes. (msb=most significant byte, lsb = *least significant byte): *) msb := and_mask(shr(intVal, 8), 16#00ff); lsb := and_mask(intVal, 16#00ff); (* Build the message: *) message := '\$00' + char(lsb) + char(msb) + '\$FF'; (* Send 4 byte message out COM5: *) result := ComWrBts(5, message, 4); </pre>

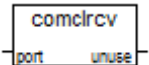
Write A Message To The Transmit Buffer -- ComWrStr()

Type	C Function
Function	Writes a string of characters to a serial port.
Syntax	ANA ComWrStr(ANA port, MSG string);
FBD	
Parameters	<p>port: which COM port to write to 1-7</p> <p>string: message containing non-null ASCII characters</p>
Return Value	error: ANA 0 = success, nonzero = failure
Remarks	Before using this function, you must open the port with ComOpen. Unlike the function ComWrBts, ComWrStr will stop if it gets to an embedded null (\$00) character. Thus if you want to send messages containing null characters, you must use ComWrBts or ComWrBt (for a single byte). ComWrBts requires you to specify the length of the message, whereas ComWrStr does not.
ST Example	<pre>(* write the string 'abcdefg' to COM3: *) result := ComWrStr(3, 'abcdefg');</pre>

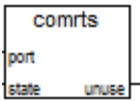
Detect When The Transmit Buffer Is Empty -- ComXmtEm()

Type	C Function
Function	Reads the transmit empty state of a serial port.
Syntax	BOO ComXmtEm(ANA port);
FBD	
Parameters	port: ANA which COM port to check 1-7
Return Value	empty: BOO transmit empty state of the port: TRUE = empty, FALSE = not empty
Remarks	It is sometimes necessary to determine when a serial transmission has completed in order to control handshaking signals or perform other tasks. This function allows you to do so. When you send serial data using the ComWrBt, ComWrBts or ComWrStr functions, the data is temporarily stored in a transmit buffer and the function returns immediately. An interrupt service routine then sends each byte in the background. When this process is complete and all the data has been transmitted, ComXmtEm will return TRUE. If the data is still being transmitted, ComXmtEm will return FALSE.
ST Example	<pre>(* is COM4 done sending? *) if ComXmtEm(4) then ... end_if; .</pre>

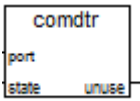
Clear The Receive Buffer -- ComClRcv()

Type	C Function
Function	Clears any bytes from the receive buffer of a serial port.
Syntax	ANA ComClRcv(ANA port);
FBD	
Parameters	port: ANA which COM port to clear 1-7
Return Value	unused
Remarks	Typical use would be to discard remaining input in order to resynchronize.
ST Example	<pre>(* clear COM3: *) result := ComClRcv(3);</pre>

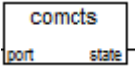
Control RTS On The ComPort -- ComRts()

Type	C Function
Function	Controls the RTS (Request To Send) signal of a serial port.
Syntax	ANA ComRts(ANA port, BOO state);
FBD	
Parameters	<p>port: ANA which COM port to control 1-7</p> <p>state: BOO desired state of RTS FALSE (inactive), TRUE (active)</p>
Return Value	unused
Remarks	The RTS signal is sometimes used for handshaking or control of external equipment, such as a radio transceiver. The RTS signal is internally connected to the transmit enable signal for RS-485 ports. For these ports, a TRUE (active) state selects transmit mode, and a FALSE (inactive) state selects receive mode. On current hardware, all ports except COM1 support RTS. See the Hardware Reference Guide for details on your controller.
ST Example	<pre>(* make COM2 RTS active *) result := ComRts(2, TRUE); ... (* make COM2 RTS inactive *) result := ComRts(2, FALSE);</pre>

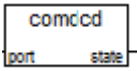
Control DTR On The ComPort -- ComDtr()

Type	C Function
Function	Controls the DTR (Data Terminal Ready) signal of a serial port.
Syntax	ANA ComDtr(ANA port, BOO state);
FBD	
Parameters	<p>port: ANA which COM port to control 1-7</p> <p>state: BOO desired state of DTR FALSE (inactive), TRUE (active)</p>
Return Value	unused
Remarks	The DTR signal is sometimes used for handshaking or control of external equipment. On current hardware. See the Hardware Reference Guide for details on your controller.
ST Example	<pre>(* make COM2 DTR active *) result := ComDtr(2, TRUE); (* make COM2 DTR inactive *) result := ComDtr(2, FALSE);</pre>

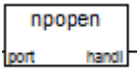
Read CTS From The ComPort -- ComCts()

Type	C Function
Function	Reads the CTS (Clear To Send) signal of an ICL-4300 serial port.
Syntax	BOO ComCts(ANA port);
FBD	
Parameters	port: ANA which COM port to access 1-7
Return Value	state: BOO state of the CTS signal: FALSE=inactive, TRUE=active
Remarks	The CTS signal is sometimes used for handshaking with external equipment. Most serial ports support CTS. See the Hardware Reference Guide for details on your controller.
ST Example	<pre>state := ComCts(3); (* read CTS from COM3 *) if state = FALSE then ... end_if;</pre>

Read DCD From The ComPort -- ComDcd()

Type	C Function
Function	Reads the DCD (Data Carrier Detect) signal of an ICL-4300 serial port.
Syntax	BOO ComDcd(ANA port);
FBD	
Parameters	port: ANA which COM port to access 1-7
Return Value	state: BOO state of the DCD signal: FALSE=inactive, TRUE=active
Remarks	The DCD signal is sometimes used for handshaking with external equipment. On current hardware, only COM2, COM5, COM6 and COM7 support DCD.
ST Example	<pre>state := ComDcd(6); (* read DCD from COM6 *) if state = TRUE then ... end_if;</pre>

Open A Network Port By Name -- NPOpen()

Type	C Function
Function	Opens an net port for reading/writing.
Syntax	ANA NPOpen(MSG port);
FBD	
Parameters	port: ANA valid net port name from config file, identifies the file to open
Return Value	handl: ANA Handle to be used to access net port, 0 on failure.
Remarks	NPOpen must be called first before a net port can be written using the NPPktSnd function. The settings associated with the net port record, as defined in Scadabuilder, determine how the port is opened. One advantage of using this function, in conjunction with NPPktSnd is that it can give you automatic control of the RTS signal. If the RTS mode is set to 'xmtOn' in Scadabuilder, and a lead and trail delay are specified, the RTS signal will be activated while the packet is sent, with the appropriate delays applied. There are no special receive functions based on the net port handle: use the normal COM receive functions (ComRdBt and ComRdBts).

Example

In ScadaBuilder **Communications | Network Ports...**:

Network Ports

Com1 Com3 Com4 Com5 Modem

Device name:

Name:

Radio Port

Serial parameters:

Baud:

9600

Data bits:

8

Stop bits:

1

Parity:

none

RTS Control:

☐ Always on ☐ Always off ☒ Transmit on

Transmit delays (ms):

20 Lead 5 Trail

Media Ready Mode:

☒ Always ready ☐ Time based

Receiver quiet time:

100 (ms)

Media access delay (ms):

0 Min 50 Max

in a Structured Text file:

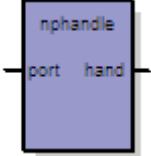
(open the net port *)*

handle := NPOpen('Radio Port');

(send a message packet *)*

result := NPPktSnd(handle, message, mlen(message));

Get A Netport Handle By Name -- NPHandle()

Type	C Function
Function	Get a Netport handle by name.
Syntax	DINT NPHandle(STRING portName);
FBD	
Parameters	portName: STRING valid net port name
Return Value	hand: DINT nHandle to be used to access net port, 0 on failure.
Remarks	<p>This function retrieves a handle that references the netport.</p> <p>The handle can be used for both the NPClose and NPPktSnd functions.</p> <p>There are no receive functions that use the handle.</p>

Example

In ScadaBuilder **Communications | Network Ports...**:

Network Ports

Com1 Com3 Com4 Com5 Modem

Device name:

Name: Radio Port

Serial parameters:

Baud: 9600

Data bits: 8

Stop bits: 1

Parity: none

RTS Control:

☐ Always on ☐ Always off ☒ Transmit on

Transmit delays (ms): 20 Lead 5 Trail

Media Ready Mode:

☒ Always ready ☐ Time based

Receiver quiet time: 100 (ms)

Media access delay (ms): 0 Min 50 Max

in a Structured Text file:

(* open the net port *)

```
handle := NPHandle( 'Radio Port');
```

(* send a message packet *)

```
result := NPPktSnd( handle, message, mlen( message ) );
```

Send A Packet Over A Network Port -- NPPktSnd()

Type	C Function
Function	Sends a packet of bytes using a net port.
Syntax	ANA NPPktSnd(ANA port, MSG bytes, ANA len);
Parameters	<p>port: ANA valid net port handle returned from NPOpen function</p> <p>bytes: MSG the message to transmit</p> <p>len: ANA the length of the message (number of bytes)</p>
Return Value	error: ANA 0 = success, nonzero = failure
Remarks	<p>Before using this function, you must open the net port with NPOpen. NPOpen returns a handle, that is used to identify the port. If len is out of range, or the net port handle is 0, nothing will be sent, and the function will return failure.</p> <p>The settings associated with the net port record, as defined in the configuration file, determine how the packet is sent. One advantage of using this function, instead of ComWrBts, is that it can give you automatic control of the RTS signal. If the RTS mode is set to 'xmtOn' in the configuration file, and a lead and trail delay are specified, the RTS signal will be activated while the packet is sent, with the appropriate delays applied.</p>

ExampleIn ScadaBuilder **Communications | Network Ports...**:

Network Ports

Com1 Com3 Com4 Com5 Modem

Device name:

Name: Radio Port

Serial parameters:

Baud: 9600

Data bits: 8

Stop bits: 1

Parity: none

RTS Control:

☐ Always on ☐ Always off ☒ Transmit on

Transmit delays (ms): 20 Lead 5 Trail

Media Ready Mode:

☒ Always ready ☐ Time based

Receiver quiet time: 100 (ms)

Media access delay (ms): 0 Min 50 Max

in a Structured Text file:

(* open the net port *)

handle := NPOpen('Radio Port');

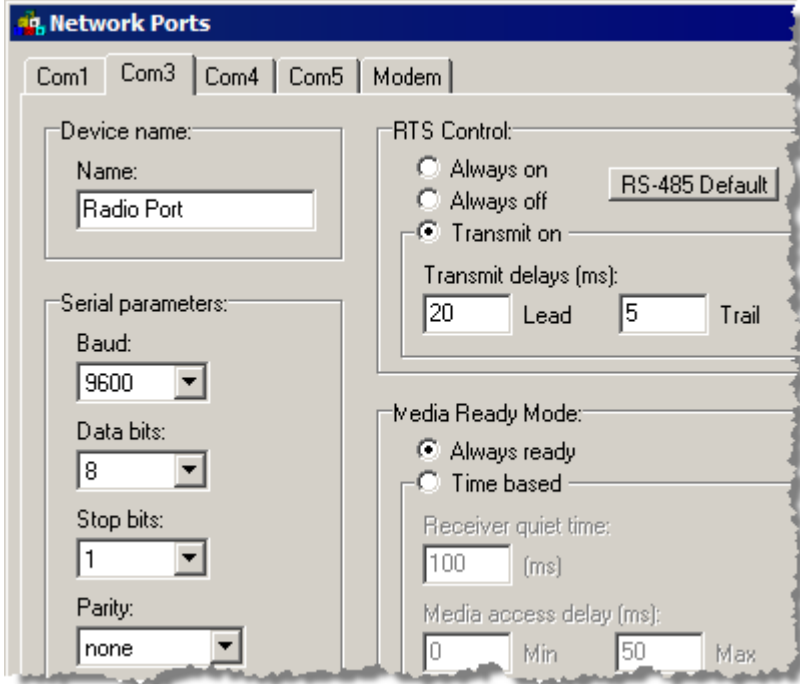
(* Build the message: *)

message := '\$00' + '\$01' + '\$02' + '\$03' + '\$04';

(* Send message out the net port *)

result := NPPktSnd(handle, message, mlen(message));

Close A Network Port By Handle -- NPClose()

Type	C Function
Function	Closes a net port that was opened with the NPOpen function.
Syntax	ANA NPClose(ANA port);
Parameters	port: ANA valid net port handle returned from NPOpen function
Return Value	error: ANA 0 = success, nonzero = failure
Remarks	This function may be called when a process is done using the net port. The handle that is used to close the port is returned from the NPOpen function. Net ports are defined in the configuration file.
Example	<p>In ScadaBuilder Communications Network Ports...:</p>  <p>in a Structured Text file:</p> <pre>(* open the net port *) handle := NPOpen('Radio Port'); ... (* close the net port *) result := NPClose(handle);</pre>

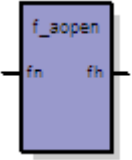
File I/O Functions

Supplemental file functions are provided to allow you to perform operations not supported by the base level ISaGRAF file functions.

For Pinnacle controllers, drives may be specified by device in the following format:

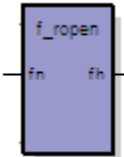
- "C:" (default)
- "IDE:"
- "USB1:"
- "USB2:"
- "USB3:"
- "USB4:"

Open a File For Appending -- f_aopen()

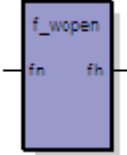
Type	C Function
Function	Opens a binary file in append mode.
Syntax	DINT f_aopen(STRING filename);
DBS	
Parameters	filename: STRING name of file to open. May include the access path to the file using the \ or / symbol to specify a directory. To ease application portability, / and \ are equivalent.
Return Value	fn: DINT file number, -1 = failure, non-zero = success
Remarks	<p>One file cannot be opened twice, a system log entry will be generated.</p> <p>If the file already exists, subsequent writes to the file will append to the information that is already there. If the file does not exist, it will be created.</p> <p>This function is similar to f_wopen, which opens the file for writing, but overwrites any existing data.</p> <p>Drives may be specified by device in the following format:</p> <ul style="list-style-type: none">"C:""IDE:""USB1:""USB2:""USB3:""USB4:"

<i>ST Example</i>	<pre>(* open the file "data.log" for appending *) result := f_aopen('c:\data.log');</pre>
-------------------	---

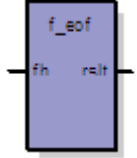
Open A Binary File For Read Only Access -- f_ropen()

<i>Type</i>	C Function
<i>Function</i>	Opens a binary file in read-only mode.
<i>Syntax</i>	DINT f_ropen(STRING filename);
<i>FBD</i>	
<i>Parameters</i>	<p>filename: STRING name of file to open.</p> <p>May include the access path to the file using the \ or / symbol to specify a directory. To ease application portability, / and \ are equivalent.</p>
<i>Return Value</i>	fn: DINT file number, -1 = failure, non-zero = success
<i>Remarks</i>	<p>One file cannot be opened twice, a system log entry will be generated.</p> <p>If the file already exists, subsequent writes to the file will append to the information that is already there. If the file does not exist, it will be created.</p> <p>This function is similar to f_wopen, which opens the file for writing, but overwrites any existing data.</p> <p>Drives may be specified by device in the following format:</p> <p>"C:"</p> <p>"IDE:"</p> <p>"USB1:"</p> <p>"USB2:"</p> <p>"USB3:"</p> <p>"USB4:"</p>
<i>ST Example</i>	<pre>filehandle:= f_ropen('ide:\data.bin'); if filehandle = 0 then errormessage := 'could not open data log file'; else myvalue := fa_read(); f_close (filehandle); end_if;</pre>

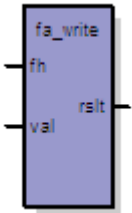
Open A Binary File In Read-Write Mode -- f_wopen()

<i>Type</i>	C Function
<i>Function</i>	Opens a binary file in read-only mode.
<i>Syntax</i>	DINT f_wopen(STRING filename);
<i>FBD</i>	
<i>Parameters</i>	<p>filename: STRING name of file to open</p> <p>May include the access path to the file using the \ or / symbol to specify a directory. To ease application portability, / and \ are equivalent.</p>
<i>Return Value</i>	fn: DINT file number, -1 = failure, non-zero = success
<i>Remarks</i>	<p>One file cannot be opened twice, a system log entry will be generated.</p> <p>If the file already exists, subsequent writes to the file will overwrite the information that is already there. If the file does not exist, it will be created.</p> <p>Drives may be specified by device in the following format:</p> <p>"C:"</p> <p>"IDE:"</p> <p>"USB1:"</p> <p>"USB2:"</p> <p>"USB3:"</p> <p>"USB4:"</p>
<i>ST Example</i>	<pre> filehandle:= f_wopen('ide:\data.bin'); if filehandle = 0 then errormessage := 'could not open data.bin file'; else result := fa_write(filehandle, myvalue); f_close (filehandle); end_if; </pre>


Check For End Of File Status -- f_eof()

Type	C Function
Function	ISaGRAF Legacy Function. Check for the end of a file while reading or writing.
Syntax	BOOL F_EOF(DINT fh);
FBD	
Parameters	fh: DINT File handle from f_Xopen() command.
Return Value	rslt: BOOL Returns TRUE if end of the file has been reached.
Remarks	
ST Example	<pre> filehandle:= fa_wopen('ide:\data.bin'); if filehandle = 0 then errormessage := 'could not open data.bin file'; (* loop ... *) value := fa_read(filehandle); if f_eof(fh) then f_close (filehandle); (* ...end loop *) end_if; (* ... end loop *) end_if; </pre>

Write An Integer To A File -- fa_write()

Type	C Function
Function	IsaGraf legacy function Write an integer to a file one at a time.
Syntax	BOOL FA_WRITE(DINT fh, DINT val);
FBD	
Parameters	fh: DINT File handle from f_wopen() or f_aopen(). val: DINT Value to write.
Return Value	rslt: BOOL TRUE on success, FALSE on a failure.
Remarks	File must be opened for writing first. Read-only opened files will fail.
ST Example	<pre> filehandle:= f_wopen('c:\data.bin'); if filehandle = 0 then errormessage := 'could not open data.bin file'; else rslt := fa_write(filehandle, myvalue); f_close (filehandle); end_if; </pre>

Write A Message Register to a Line in a Text File -- fm_writecrlf()

Type	C Function
Function	IsaGraf legacy function Write CRLF delimited Message registers to a file one at a time for viewing in a text editor or reading back with the fm_readcrlf().
Syntax	BOOL FM_WRITECRLF(DINT fh, STRING str);
FBD	
Parameters	fh: DINT File handle from f_wopen() or f_aopen(). str: MSG Value to write.
Return Value	rslt: BOOL TRUE on success, FALSE on a failure.
Remarks	<p>File must be opened for writing or appending first. Read-only files will fail.</p> <p>CR = Carriage Return '\$0A' is a CR string in ISaGRAF '\r' is a CR in many editors and C programs</p> <p>LF = Line Feed '\$0D' is a LF string in ISaGRAF '\n' is a LF string in many editors and C programs.</p> <p>Each message register written will be terminated with a CRLF.</p> <p>For instance, if the string 'My File Line' is written, when opened, the file will contain: "My File Line \r\n"</p> <p>See <i>Read A Line From a Text File to a Message Register -- fm_readcrlf()</i> (on page 537) for the reciprocal read function.</p>

ST Example

```
(* open the file for writing *)
fh1 := f_wopen('ide:MyCfg.ini');
if fh1 = 0 then
    testfail := true;
    teststat := 'f_wopen MyCfg.ini failed!';
end_if;

(* write up to 600 lines from message registers *)
for i := 501 TO 600 DO

    (* get the string data by index *)
    msg := msgrd( i, 1,255 );

    (* if there is any data in the string *)
    if ( mlen( msg ) > 0 ) then

        (* write the line to the file *)
        fRslt := fm_writecrlf( fh1, msg );

        (* check the result to see if the write was successful *)
        if not fRslt then
            testfail := true;
            teststat := 'fm_writecrlf() MyCfg.ini failed!';
            return;
        end_if;

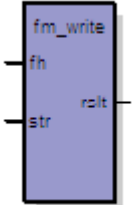
    end_if;

end_for;

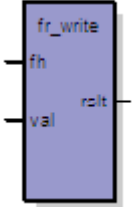
(* test normal close function *)
fRslt := f_close( fh1 );

if not fRslt then
    testfail := true;
    teststat := 'f_close MyCfg.ini failed!';
else
    fh1 := 0;
end_if;
```

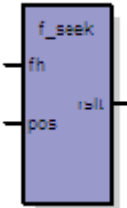
Write A Message Register (STRING) To A File -- fm_write()

Type	C Function
Function	IsaGraf legacy function Write Message register to a file one at a time for later retrieval using the fm_read() function.
Syntax	BOOL FM_WRITE(DINT fh, STRING str);
FBD	
Parameters	fh: DINT File handle from f_wopen() or f_aopen(). str: STRING Value to write.
Return Value	rslt: BOOL TRUE on success, FALSE on a failure.
Remarks	File must be opened for writing or appending first. Read-only files will fail. Each message register written will be terminated by a NULL to set it apart from other registers. The same delimiter is used when reading the file. See Read A Message Register (STRING) From A File -- fm_read() (on page 539), This function should not be used to make files that will be viewed in text editors later. See Write A Message Register to a Line in a Text File -- fm_writecrlf() (on page 531) if this is the intent for writing the data to a file.
ST Example	<pre> filehandle:= f_wopen('c:\data.bin'); if filehandle = 0 then errormessage := 'could not open data.bin file'; else rslt := fm_write(filehandle, myString); f_close (filehandle); end_if; </pre>

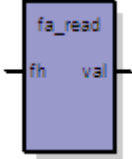
Write A Real Register To A File -- fr_write

Type	C Function
Function	IsaGraf legacy function Write real number to a file one at a time.
Syntax	BOOL FR_WRITE(DINT fh, REAL val);
FBD	
Parameters	fh: DINT File handle from f_wopen() or f_aopen(). val: REAL Value to write.
Return Value	rslt: BOOL TRUE on success, FALSE on a failure.
Remarks	File must be opened for writing or appending first. Read-only files will fail.
ST Example	<pre> filehandle:= f_wopen('c:\data.bin'); if filehandle = 0 then errormessage := 'could not open data.bin file'; else rslt := fa_rrite(filehandle, myvalue); f_close (filehandle); end_if; </pre>

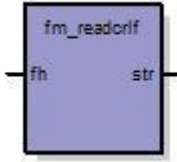
Seek Or Get The Position Of A File -- f_seek()

Type	C Function
Function	Seek a postion by byte in an existing file.
Syntax	DINT F_SEEK(DINT fh, DINT pos);
FBD	
Parameters	<p>fh: DINT File handle from f_Xopen() command</p> <p>pos: DINT</p> <p>a positive sets the file pointer to the position and the position is returned. a 0 sets the file pointer to beginning of the file and 0 is returned on success. -1 sets the file pointer to the end of the file and returns the file size. -2 moves the file point from the current position and returns the new position. -3 returns the current position without modifying it.</p>
Return Value	<p>rslt: DINT See pos input for details.</p> <p>If there is an error or a bad parameter, then a -1 is returned.</p>
Remarks	None.
ST Example	<pre> filehandle:= f_wopen('ide:\data.dat'); if filehandle = 0 then errormessage := 'could not open data.bin file'; else pos = f_seek(fileHandle, 25) if pos > 0 then value := fa_read(filehandle); end_if; f_close (filehandle); end_if; </pre>

Read An Integer Register From A File -- fa_read()

Type	C Function
Function	IsaGraf legacy function Read an Integer Value from a File.
Syntax	DINT FA_READ(DINT fh);
FBD	
Parameters	fh: DINT file handle from f_Xopen()
Return Value	val: DINT integer value from file. Returns 0 on a failure.
Remarks	None.
ST Example	<pre> filehandle:= f_ropen('ide:\data.bin'); if filehandle = 0 then errormessage := 'could not open data.bin file'; else value := fa_read(filehandle); f_close (filehandle); end_if; </pre>

Read A Line From a Text File to a Message Register -- fm_readcrlf()

Type	C Function
Function	IsaGraf legacy function Read a CR, LF or CRLF delimited line of a file to a String (Message).
Syntax	STRING FM_READCRLF(DINT fh);
FBD	
Parameters	fh: DINT file handle from f_Xopen()
Return Value	str: STRING series of bytes as a string from a file. If there is an error reading the file (such as the end has been reached) then the string '<read error>' is returned in the target Message register.
Remarks	<p>CR = Carriage Return '\$0A' is a CR string in ISaGRAF '\r' is a CR in many editors and C programs</p> <p>LF = Line Feed '\$0D' is a LF string in ISaGRAF '\n' is a LF string in many editors and C programs.</p> <p>If a file line is read with CR delimiter, LF delimiter or a CRLF delimiter such as: "My file line\r", "My file line\n" or "My file line\r\n" respectively; the string returned to the message register is 'My file line'.</p> <p>The file position is always left at the next character after the '\r', '\n', or '\r\n' unless the end of the file has been reached.</p> <p>To write data to text file line by line to be read by fm_readcrlf(), see Write A Message Register to a Line in a Text File -- fm_writecrlf() (on page 531)</p>

ST Example

```
(* open the file *)
fh1 := f_ropen('ide:MyCfg.ini');
if fh1 = 0 then
    testfail := true;
    teststat := 'f_ropen MyCfg.ini failed!';
    return;
end_if;

(* read up to 100 message registers from a file *)
for i := 501 TO 600 DO

    (* check to see if we are at the end of the file *)
    if not f_eof( fh1 ) then

        (* write up to 255 bytes into the local message register *)
        iRslt := msgwr( i, 1, fm_readcrlf( fh1 ), 255 );

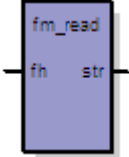
    end_if;

end_for;

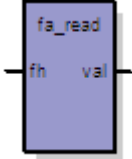
(* close the file *)
fRslt := f_close( fh1 );
if not fRslt then
    testfail := true;
    teststat := 'f_close MyCfg.ini failed!';
else
    fh1 := 0;
end_if;

(* open the file for writing *)
fh1 := f_wopen('ide:MyCfg.ini');
if fh1 = 0 then
    testfail := true;
    teststat := 'f_wopen MyCfg.ini failed!';
end_if;
```

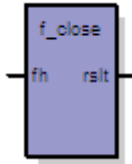
Read A Message Register (STRING) From A File -- fm_read()

Type	C Function
Function	IsaGraf legacy function Read an String (Message) from a File
Syntax	STRING FM_READ(DINT fh);
FBD	
Parameters	fh: DINT file handle from f_Xopen()
Return Value	str: STRING series of bytes as a string from a file.
Remarks	None.
ST Example	<pre> filehandle:= f_ropen('c:\data.txt'); if filehandle = 0 then errormessage := 'could not open data.bin file'; else myString := fm_read(filehandle); f_close (filehandle); end_if; </pre>

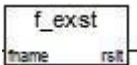
Read A Real Value From A File -- fr_read();

Type	C Function
Function	IsaGraf legacy function Read Real Value from a File.
Syntax	REAL FR_READ(DINT fh);
FBD	
Parameters	fh: DINT file handle from f_Xopen()
Return Value	val: REAL floating point value from file. Returns 0.0 on a failure.
Remarks	None.
ST Example	<pre> filehandle:= f_ropen('ide:\data.bin'); if filehandle = 0 then errormessage := 'could not open data.bin file'; else value := fr_read(filehandle); f_close (filehandle); end_if; </pre>

Close A File -- f_close()

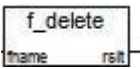
<i>Type</i>	C Function
<i>Function</i>	Opens a binary file in read-only mode.
<i>Syntax</i>	BOOL f_close(DINT fh);
<i>FBD</i>	
<i>Parameters</i>	fh: DINT File handle from f_open
<i>Return Value</i>	rslt: BOOL TRUE if successful. False if there was a failure.
<i>Remarks</i>	Not closing a file after writes are done could cause data loss if there is a loss of power. All files are closed on an application shutdown.
<i>ST Example</i>	<pre> filehandle:= f_ropen('c:\data.bin'); if filehandle = 0 then errormessage := 'could not open data.bin file'; else value := fa_read(filehandle); result := f_close (filehandle); end_if; </pre>

Check To See If A File Exists -- f_exist()

Type	C Function
Function	Checks for the existence of a file on the ICL controller.
Syntax	ANA f_exist(MSG filename);
FBD	
Parameters	<p>fname: name of the file to check.</p> <p>May include the access path to the file using the \ or / symbol to specify a directory. To ease application portability, '/' and '\ ' are equivalent.</p> <p>Wildcards may be specified for multiple file checks. Specifying "*" as a file name will check for any file in the specified directory. Specifying a *.<ext> will look for any file with that extension. For example:</p> <pre>fname = 'ide:\logs*.lg1'</pre> <p>This specification will find .lg1 files in the logs directory on the IDE drive.</p> <p>'?' may also be used to specify a single character wild card in the name. For example:</p> <pre>fname = 'usb1:\logs*.lg?'</pre> <p>This specification will check for any files with the .lg extension including .lg1 and lg2 files in the directory 'logs' on the usb drive.</p> <p>Drives may be specified by device in the following format:</p> <pre>"C:" "IDE:" "USB1:" "USB2:" "USB3:" "USB4:"</pre> <p>If the drive is not specified, c:\ or ide:\ is assumed.</p>
Return Value	rslt: ANA 0 = exist, -1 = does not exist.
Remarks	It is suggested that this function only be called when needed. This function will wait for all other disk system operations to complete before activating thus suspending the PLC cycle during that time.

ST Example	<pre>(* if the check now flag is set *) if fCheckNow then (* call the check function to see if the file exists *) result := f_exist('c:\data.log'); (* reset the check flag *) fCheckNow := false; end_if; (* check result and post error *) if result = -1 then (* annunciate the result *) msgError := 'file not found'; end_if;</pre>
-------------------	---

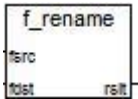
Delete A File -- f_delete()

<i>ype</i>	C Function
<i>Function</i>	Checks for the existence of a file on the ICL controller.
<i>Syntax</i>	ANA f_delete(MSG filename);
<i>FBD</i>	

<i>Parameters</i>	<p>fname: MSG name of file to open.</p> <p>May include the access path to the file using the \ or / symbol to specify a directory. To ease application portability, '/' and '\' are equivalent.</p> <p>Wildcards may be specified for multiple file deletes. Specifying "*.*)" as a file name will delete all files in the specified directory. Specifying a *.<ext> will delete any file with that extension. For example:</p> <p>fname = 'ide:\logs*.lg1'</p> <p>This specification will delete all .lg1 files in the logs directory of the ide drive.</p> <p>'?' may also be used to specify a single character wild card in the name. For example:</p> <p>fname = 'usb1:\logs*.lg?'</p> <p>This specification will delete all .lg1 and lg2 files to the directory 'logs' on the usb drive.</p> <p>Drives may be specified by device in the following format:</p> <p>"C:"</p> <p>"IDE:"</p> <p>"USB1:"</p> <p>"USB2:"</p> <p>"USB3:"</p> <p>"USB4:"</p> <p>If the drive is not specified, c:\ or ide:\ is assumed.</p>
<i>Return Value</i>	<p>rslt: ANA 0 = success, -1 = failure.</p>
<i>Remarks</i>	<p>It is suggested that this function only be called only when needed. This function will wait for all other disk system operations to complete before activating thus suspending the PLC cycle during that time.</p> <p>Care should be taken to only use wildcards in sub folders. An fname of ide:*.*) will delete all system file in the root directory of the controller. A complete controller setup... will be required to get the program files back on the unit.</p>


ST Example	<pre>(* if the delete now flag is set *) if fDeleteNow then (* attempt to delete the file *) result := f_delete('c:\data.log'); (* reset the delete flag *) fDeleteNow := false; end_if; (* check result and post error *) if result = -1 then (* announce the result *) msgError := 'file not found or read-only'; end_if;</pre>
------------	---

Rename A File -- f_rename()

Type	C Function
Function	Renames a file.
Syntax	ANA f_rename(MSG src, MSG dst);
FBD	
Parameters	<p>src: MSG name of source file to rename</p> <p>May include the access path to the file using the '\' or '/' symbol to specify a directory. To ease application portability, '/' and '\' are equivalent.</p> <p>dst: name and path of destination file</p> <p>The same special characters may be used.</p> <p>"C:"</p> <p>"IDE:"</p> <p>"USB1:"</p> <p>"USB2:"</p> <p>"USB3:"</p> <p>"USB4:"</p> <p>If the drive is not specified, c:\ or ide:\ is assumed.</p>
Return Value	rslt: ANA 0 = success, -1 = failure.
Remarks	It is suggested that this function only be called when needed. This function will wait for all other disk system operations to complete before activating thus suspending the PLC cycle during that time.

ST Example	<pre>(* if the check now flag is set *) if fRenameNow then (* call the check function to see if the file exists *) result := f_rename('c:\data.log', 'c:\data.csv'); (* reset the check flag *) fRenameNow := false; end_if; (* check result and post error *) if result <> 0 then (* annunciate the result *) msgError := 'error renaming data.log file'; end_if;</pre>
------------	--

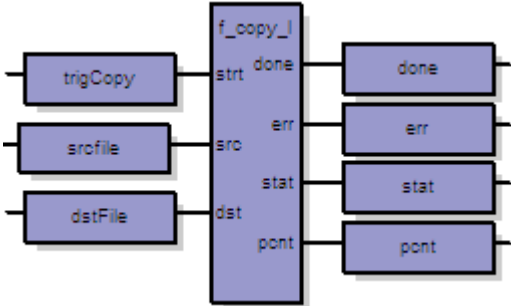
Copy A File -- f_copy()

Type	C Function
Function	Copies a file.
Syntax	DINT f_copy(MSG src, MSG dst);
FBD	

<i>Parameters</i>	<p>src: STRING May include the access name of source file path to the file using to copy the \ or / symbol to specify a directory. To ease application portability, / or \ are equivalent.</p> <p>Wildcards may be specified for multiple file copies. Specifying "*" as a fsrc file name will copy all files in the source directory. Specifying a *.<ext> will copy any file with that extension. For example:</p> <pre>src = 'ide:\logs*.lg1'</pre> <pre>dst = 'usb1:*.*' or 'usb1:\' or just 'usb1:'</pre> <p>This specification will copy all .lg1 files in the logs directory to the destination usb drive.</p> <p>'?' may also be used to specify a single character wild card in the name. For example:</p> <pre>src = 'ide:\logs*.lg?'</pre> <pre>dst = 'usb1:\logs*.*' or 'usb1:\logs\' or just 'usb1:\logs'</pre> <p>This specification will copy all .lg1 and lg2 files to the destination directory 'logs' on the usb drive. Destination directories MUST be created before the copy operation or the copy will fail in this case.</p> <p>dst: STRING same as src argument name and path of destination file. When wildcard characters '*' or '?' are used in the src specification the file name portion of the dst is ignored and only drive and directory are used for the destination. The file names themselves come from those found in the src file specification.</p> <p>Drives may be specified by device in the following format:"C:"</p> <pre>"IDE:"</pre> <pre>"USB1:"</pre> <pre>"USB2:"</pre> <pre>"USB3:"</pre> <pre>"USB4:"</pre> <p>If the drive is not specified, c:\ or ide:\ is assumed.</p>
<i>Return Value</i>	result, 0 = success, -1 = failure.
<i>Remarks</i>	<p>It is suggested that this function only be called when needed.</p> <p>The result reflects that initialization was carried out but NOT that the file copy itself was finished. See status lights on drives to be sure the copy is done or use the f_exist() function to verify that the destination file is done.</p> <p>If you need more status on the copy process, use the f_copy_l() function block.</p> <p>This function is available for Pinnacle controllers only.</p>

ST Example	<pre>(* if the check now flag is set *) if fCopyNow then (* call the check function to see if the file has been successfully copied*) result := f_copy('c:\data.log', 'c:\data.csv'); (* reset the check flag *) fCopyNow := false; end_if; (* check result and post error *) if result <> 0 then (* announce the result *) msgError := 'error renaming data.log file'; end_if;</pre>
------------	---

Copy A Large File -- f_copy_l()

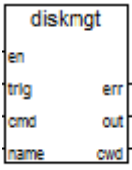
Type	C Function
Function	Copies a file on the ICL controller to another file or drive. Used for large time consuming file copies.
Syntax	DINT f_copy(MSG src, MSG dst);
FBD	

<i>Parameters</i>	<p>strt: BOOL</p> <p>TRUE = Start a copy process. Must be set to FALSE for one scan to initialize stats and start a new process on TRUE.</p> <p>FALSE = Clear the status outputs and make block ready for the next copy. FALSE is ignored until the current copy is complete.</p> <p>src: STRING May include the access name of source file path to the file using to copy the \ or / symbol to specify a directory. To ease application portability, / or \ are equivalent.</p> <p>Wildcards may be specified for multiple file copies. Specifying "*.*)" as a filename will copy all files in the source directory. Specifying a *.<ext> will copy for any file with that extension. Specifying just a '*' as the src will copy all files, even those without extensions.</p> <p>For example:</p> <pre>src = 'ide:\logs*.lg1'</pre> <pre>dst = 'usb1:.*.*' or 'usb1:\' or just 'usb1:'</pre> <p>This specification will copy all .lg1 files in the logs directory to the destination usb drive.</p> <p>'?' may also be used to specify a single character wild card in the name. For example:</p> <pre>src = 'ide:\logs*.lg?'</pre> <pre>dst = 'usb1:\logs*.*)' or 'usb1:\logs\' or just 'usb1:\logs'</pre> <p>This specification will copy all .lg1 and lg2 files to the destination directory 'logs' on the usb drive. Destination directories MUST be created or the copy will fail in this case.</p> <p>dst: STRING same as src argument name and path of destination file. When wildcard characters '*' or '?' are used in the src specification the file name portion of the dst is ignored and only drive and directory are used for the destination. The file names themselves come from those found in the src file specification.</p> <p>Drives may be specified by device in the following format:</p> <pre>'C:'</pre> <pre>'IDE:'</pre> <pre>'USB1:'</pre> <pre>'USB2:'</pre> <pre>'USB3:'</pre> <pre>'USB4:'</pre> <p>If the drive is not specified, c:\ or ide:\ is assumed.</p>
-------------------	--

<i>Return Values</i>	<p>done: BOOL TRUE = File Copy function completed or errored out.</p> <p>err : BOOL TRUE = Error during copy check stat output.</p> <p>stat: DINT</p> <p>A positive value reflects the current status as the number of bytes copied. When the copy is done, this is the total bytes copied until the strt input is set to false.</p> <p>0 = Ready to receive a start command from the strt input.</p> <p>-1 = Source file not found or access denied.</p> <p>-2 = Source file could not be opened.</p> <p>-3 = Source file information could not be obtained</p> <p>-4 = Destination file could not be opened.</p> <p>-5 = Memory for copy operation could not be allocated.</p> <p>-6 = Source file data read error.</p> <p>-7 = Destination file data write error.</p> <p>-100 = Memory allocation for copy function failed</p> <p>-101 = Source drive not found.</p> <p>-102 = Destination drive not found</p> <p>-103 = Copy aborted, unspecified error</p> <p>pcnt: DINT Progress percentage. 100% will be displayed after copy is complete until strt input is set to false.</p>
<i>Remarks</i>	<p>This function during operation can affect the ISaGRAF scan time up to 100mS per scan. Copies to USB drives take about 3 times as long as copy operations on the internal IDE drive alone. If a shutdown occurs, the files will be closed properly. The same is true on a power fail. The destination file will be a partial copy at that time however. Existing destination files will always be overwritten. It is recommended that only one copy be done at a time to limit the effects on system resources and scan time.</p>

<i>ST Example</i>	<pre>(* setting the startcopy flag to false will reset status outputs and get the function ready for the next TRUE transition of startcopy to start the next file copy operation *) (* this function must run every scan to update status outputs-this is optional filecopyL below is a function block instance of f_copy_l declared in the dictionary *) filecopyL(startcopy, src, dst); (* get the status output of the file copy function *) done := filecopyL.done; err := filecopyL.err; stat := filecopyL.stat; pcnt := filecopyL.pcnt; (* has there been an error copying the file *) If stat < 0 then (* failure code to report error here *) end_if;</pre>
-------------------	--

Check Disk Space Or Create A Directory -- diskmgmt()

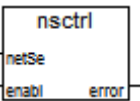
Type	C Function Block
Function	The disk management function block allows several functionalities for managing the disk of the controller. The user may create directories, change directories and examine the amount of disk space left on the drive of the controller. The current working directory is returned on any rising edge triggered command.
Syntax	ANA diskmgmt(BOO en, BOO trig, ANA cmd);
FBD	
Parameters	<p>en: BOO enable or disable block</p> <p>trig: BOO trigger activation of function selected (rising edge triggered)</p> <p>cmd: ANA command to execute when enable is true and trig goes from false to true. Invalid values are ignored.</p> <p>0 - gets the current free disk space.</p> <p>1 - creates a directory using the name.</p> <p>2 - changes directory to the named directory.</p> <p>3 - gets a file size (Pinnacle and later controllers only)</p> <p>name: name of new directory to be created or changed to.</p>
Return Value	<p>err: ANA any function error returns true</p> <p>out: ANA function output or error type</p> <p>if disk space command is selected then this returns the remaining free disk space.</p> <p>if there is a TRUE on the err output then out will return:</p> <p>2 - Bad cmd argument</p> <p>-1 - Bad folder name or error reading free diskspace</p> <p>0 - Command (make or change directory) completed</p> <p>2 - No such file or directory</p> <p>5 - Permission denied</p> <p>cwd: MSG after each operation, the current working directory will be presented at this output.</p>
Remarks	When changing directories (cmd = 2), an input of '..' will go back a directory and an input of '\' will go back to the root directory.

Cautions	<p>All names of directories follow the eight character file convention.</p> <p>No special characters are allowed except underscores '_'.</p> <p>One DISKMGT block may be used for multiple commands but the trig input must be toggled to activate the next command.</p> <p>Setting the trig input to false clears all errors.</p> <p>Downloading programs after changing directories will store the program in the current directory when doing a debugger download. Care should be used to restore the root directory.</p> <p>To use the function block in structured text , you must declare a function block instance for each operation used.</p>
ST Example	<pre> (* DiskInfo() is a function block instance of DiskMgt() *) (* read up the disk size from the controller *) DiskInfo(enable, trigger, 0, "); freeSpace := DiskInfo.out; (* create a directory called applets *) DiskInfo(enable, trigger, 1 'applets'); if not(DiskInfo.err) then(* change to the directory called applets *) DiskInfo(enable, trigger, 2 'applets'); else (* store the current error *) errorNum := DiskInfo.out; end_if; </pre>

Network Control Functions

Network communications functions provide you with control over the built-in communications protocol drivers.

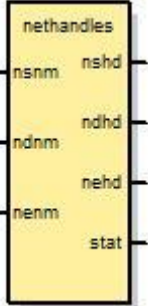
Enable Or Disable A Network Session By Name -- NSCtrl()

<i>Type</i>	C Function
<i>Function</i>	Enable or disable a network session.
<i>Syntax</i>	ANA NSCtrl(MSG netSession, BOO enable);
<i>FBD</i>	
<i>Parameters</i>	<p>netSession: MSG Name of the network session to enable/disable. Must match a network session as defined in Scadabuilder. This parameter is Case Sensitive.</p> <p>enable: BOO TRUE = enable, FALSE = disable</p>
<i>Return Value</i>	error: ANA 0 = success, nonzero = failure
<i>Remarks</i>	<p>This function is used to enable or disable a network session. A network session is used to automatically perform communications using one of the protocols built-in to the ISaGRAF kernel. You may wish to disable a network session, for example, if you want to "take over" the serial port in your application and send/receive some information. When you're done with the operation, you may then re-enable the network session.</p> <p>This function may also be used to enable or disable Dialer Call Groups to allow different Call Groups to run on difference schedules for example.</p> <p>When you disable the network session, it does not close the port.</p>

<i>ST Example</i>	<pre> (* initialization *) (* disable the network session *) result := NSCtrl('ModbusModemLink', FALSE); (* open the modem port for manual operations *) handle := NPOpen('ModbusModemPort'); ComDtr(7, TRUE); (* turn on DTR to enable modem *) (* send command to modem to enable auto-answer mode *) result := NPPktSnd(handle, 'ATS0=1\$r', 7); . (* close the modem port *) NPClose(handle); (* end of initialization *) ----- (* monitor modem carrier detect for a connection to be * established *) if online = FALSE and ComDcd(7) = TRUE then (* enable the network session *) result := NSCtrl('ModbusModemLink', TRUE); online := TRUE; end_if; (* monitor modem carrier detect for disconnect *) if online = TRUE and ComDcd(7) = FALSE then (* disable the network session *) result := NSCtrl('ModbusModemLink', FALSE); online := FALSE; end_if; </pre>
-------------------	--

Retrieve NetEvent, NetDest, NetSession Handles -- nethandles()

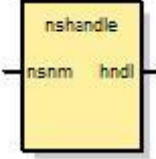
<i>Type</i>	C Function Block
<i>Function</i>	Retrieve a Network Session, Network Destination, and Network Event handles by name as configured from ScadaBuilder.

<i>Syntax</i>	<p>nethandles(STRING NetworkSessionName, STRING NetworkDestinationName, STRING NetworkEventName);</p> <p>Returns:</p> <p>DINT netHandles, networkSessionHandle</p> <p>DINT netHandles.networkDestinationHandle</p> <p>DINT netHandles.networkEventHandle</p>
<i>FBD</i>	
<i>Parameter</i>	<p>nsnm - NetworkSessionName STRING(255) Name of the Network Session as configured in ScadaBuilder. This string input is case sensitive.</p> <p>ndnm - NetworkDestinationName STRING(255) Name of the Network Destination as configured in ScadaBuilder. This string input is case sensitive.</p> <p>nenm - NetworkEventName STRING(255) Name of the Network Event as configured in ScadaBuilder. This string input is case sensitive.</p>
<i>Return Value</i>	<p>nshd - DINT returns a non zero 32 bit handle to be used with ndhandle(). If no Network Session by the string name is found then a zero is returned.</p> <p>ndhd - DINT returns a non zero 32 bit handle to be used with nehandle() function. If no Network Destination by the string name is found then a zero is returned.</p> <p>nehd - DINT returns a non zero 32 bit handle to be used with netrigger() and nepending(). If no Network Event by the string name is found then a zero is returned.</p> <p>stat - DINT returns a zero if all handles were found.</p> <p>0 = All handles found.</p> <p>1 = Could not find Network Session by name.</p> <p>2 = Could not find Network Destination by name.</p> <p>3 = Could not find Network Event by name.</p>

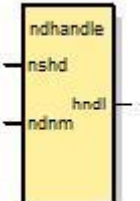
<i>Remarks</i>	<p>Handles must be renewed every time the application starts up as they can move around in memory from restart to restart (especially during development).</p> <p><i>DO NOT STORE HANDLES TO RETAINED VARIABLES.</i></p> <p>For program efficiency, it is recommended that you store handles to a registers and reuse the handles for a runtime instance rather than renewing handles each time they are used. This is especially true of you have large numbers of Network Destinations and Network Events.</p> <p>The function block nethandles() is less efficient than the discrete nshandle(), ndhandle(), and nehandle() functions though it does the same job. It has to execute all name lookups every time it is run. For a few Network Events and Network Destinations, this is fine. Larger applications with lots of Network Destinations and Network Events should use the individual functions which are more efficient doing as few lookups as possible.</p>
<i>ST Example</i>	<pre> (* Only initialize once *) if init then init := true; (* gethandles() is a function block instance of nethandles() *) (* Network Session, Network Destination, Network Event *) gethandles(' ModbusMaster', 'Slave1', 'Slave1NE1'); ghstatus := gethandles.status; (* if there was no failure *) if ghstatus = 0 then (* store off the Network Session handle *) ns := gethandles.networkSessionHandle; (* store off the Network Destination handle *) nd := gethandles.networkDestinationHandle; (* store off the Network Event handle *) netevent := gethandles.networkEventHandle; end_if; (* ghstatus = 0 *) end_if; (* init *) ... (* trigger the event *) if trigCondition and netevent <> 0 then result := netrigger(netevent); end_if; (* monitor the progress of the Network Event *) if netevent <> 0 then ne_pending := nependig(netevent); end_if; </pre>

Retrieve A Network Session Handle -- nshandle()

<i>Type</i>	C Function
<i>Function</i>	Retrieve a Network Session handle as configured from ScadaBuilder. The output of this function must be used with ndhandle().

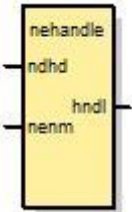
<i>Syntax</i>	DINT nshandle(STRING NetworkSessionName);
<i>FBD</i>	
<i>Parameter</i>	nsnm - NetworkSessionName STRING(255) Name of the Network Session as configured in ScadaBuilder. This string input is case sensitive.
<i>Return Value</i>	hdl - DINT returns a non zero 32 bit handle to be used with ndhandle(). If no Network Session by the string name is found then a zero is returned.
<i>Remarks</i>	<p>Handles must be renewed every time the application starts up as they can move around in memory from restart to restart (especially during development).</p> <p><i>DO NOT STORE HANDLES TO RETAINED VARIABLES.</i></p> <p>For program efficiency, it is recommended that you store the handle to a register and reuse the handle for a runtime instance rather than renewing the handle each time it is used. This is especially true of you have large numbers of Network Destinations and Network Events.</p>
<i>ST Example</i>	<pre> (* only initialize handles once *) if not init then init := TRUE; else return; end_if; (* not init *) (* get the Network Session handle, may be reused for more Network Destinations*) ns := nshandle('ModbusMaster'); if ns <> 0 then (* get the Network Destination handle, may be reused for more Network Events *) nd := ndhandle(ns, 'Slave1'); if nd <> 0 then (* get the Network Event handle, *) (* may now be used for nettrigger() and nepending() functions *) netevent := nehandle(nd, 'Slave1NE1'); end_if; (* nd <> 0 *) end_if; (* ns <> 0 *) ... (* trigger the event *) if trigCondition and netevent <> 0 then result := nettrigger(netevent); end_if; (* monitor the progress of the Network Event *) if netevent <> 0 then ne_pending := nepending(netevent); end_if; </pre>

Retrieve A Network Destination Handle -- ndhandle()

<i>Type</i>	C Function
<i>Function</i>	Retrieve Network Destination Handle as configured from ScadaBuilder. The output of this function can only be used with <i>Retrieve A Network Event Handle -- nehandle()</i> (on page 560)
<i>Syntax</i>	DINT nehandle(DINT NetworkSessionHandle, STRING NetworkDestinationName);
<i>FBD</i>	
<i>Parameters</i>	<p>nshd - NetworkSessionHandle DINT Handle from the nshandle() function. DO NOT PASS ANY OTHER PARAMETER other than a handle provided by the nshandle() or gethandles() functions.</p> <p>ndnm - NetworkDestinationName STRING(255) Name of the Network Destination as configured in ScadaBuilder. This string input is case sensitive.</p>
<i>Return Value</i>	hndl - DINT returns a non zero 32 bit handle to be used with nehandle() function. If no Network Destination by the string name is found then a zero is returned.
<i>Remarks</i>	<p>Handles must be renewed every time the application starts up as they can move around in memory from restart to restart (especially during development).</p> <p>DO NOT STORE HANDLES TO RETAINED VARIABLES.</p> <p>For program efficiency, it is recommended that you store the handle to a register and reuse the handle for a runtime instance rather than renewing the handle each time it is used. This is especially true of you have large numbers of Network Destinations and Network Events.</p>

ST Example	<pre>(* only initialize handles once *) if not init then init := TRUE; else return; end_if; (* not init *) (* get the Network Session handle, may be reused for more Network Destinations*) ns := nshandle('ModbusMaster'); if ns <> 0 then (* get the Network Destination handle, may be reused for more Network Events *) nd := ndhandle(ns, 'Slave1'); if nd <> 0 then (* get the Network Event handle, *) (* may now be used for nettrigger() and nepending() functions *) netevent := nehandle(nd, 'Slave1NE1'); end_if; (* nd <> 0 *) end_if; (* ns <> 0 *) ... (* trigger the event *) if trigCondition and netevent <> 0 then result := nettrigger(netevent); end_if; (* monitor the progress of the Network Event *) if netevent <> 0 then ne_pending := nepending(netevent); end_if;</pre>
------------	---

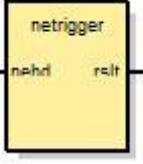
Retrieve A Network Event Handle -- nehandle()

Type	C Function
Function	Retrieve Network Event handle as configured from ScadaBuilder. The output of this function can be used with both nettrigger() and nepending() functions.
Syntax	DINT nehandle(DINT NetworkDestinationHandle, STRING NetworkEventName);
FBD	 The FBD diagram shows a yellow rectangular function block labeled 'nehandle' at the top. It has two input ports on the left: 'ndhd' (top) and 'nenm' (bottom). It has one output port on the right labeled 'hndl'.

<i>Parameters</i>	<p>ndhd - NetworkDestinationHandle DINT Handle from the ndhandle() function. DO NOT PASS ANY OTHER PARAMETER other than a handle provided by the ndhandle() or gethandles() functions.</p> <p>nenm - NetworkEventName STRING(255) Name of the Network Event as configured in ScadaBuilder. This string input is case sensitive.</p>
<i>Return Value</i>	hdl - DINT returns a non zero 32 bit handle to be used with nettrigger() and nepending(). If no Network Event by the string name is found then a zero is returned.
<i>Remarks</i>	<p>Handles must be renewed every time the application starts up as they can move around in memory from restart to restart (especially during development).</p> <p><i>DO NOT STORE HANDLES TO RETAINED VARIABLES.</i></p> <p>For program efficiency, it is recommended that you store the handle to a register and reuse the handle for a runtime instance rather than renewing the handle each time it is used. This is especially true of you have large numbers of Network Destinations and Network Events.</p>
<i>ST Example</i>	<pre> (* only initialize handles once *) if not init then init := TRUE; else return; end_if; (* not init *) (* get the Network Session handle, may be reused for more Network Destinations*) ns := nshandle('ModbusMaster'); if ns <> 0 then (* get the Network Destination handle, may be reused for more Network Events *) nd := ndhandle(ns, 'Slave1'); if nd <> 0 then (* get the Network Event handle, *) (* may now be used for nettrigger() and nepending() functions *) netevent := nehandle(nd, 'Slave1NE1'); end_if; (* nd <> 0 *) end_if; (* ns <> 0 *) ... (* trigger the event *) if trigCondition and netevent <> 0 then result := nettrigger(netevent); end_if; (* monitor the progress of the Network Event *) if netevent <> 0 then ne_pending := nepending(netevent); end_if; </pre>

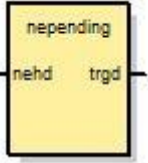
Manually Trigger A Network Event -- nettrigger()

<i>Type</i>	C Function
-------------	------------

<i>Function</i>	Trigger a Network Event by Network Event handle as configured from ScadaBuilder.
<i>Syntax</i>	DINT nehandle(DINT NetworkDestinationHandle, STRING NetworkEventName);
<i>FBD</i>	
<i>Parameters</i>	nehnd - NetworkEventHandle DINT Handle from the nehandle() function. DO NOT PASS ANY OTHER PARAMETER other than a handle provided by the nehandle() or nethandles() functions.
<i>Return Value</i>	rslt - DINT returns a zero if the Network Event was successfully triggered. Returns a -1 if a zero was passed in on nehnd input.
<i>Remarks</i>	<p>This function works as a logical OR to any Triggers that are configured in ScadaBuilder. If a Network Event is configured as a "Manual" trigger then this is the only way to Activate the Network Event. The Network Event's progress may be monitored by the nepending() function.</p> <p>This function in combination with the nepending() function can be very helpful in resolving read before write issues and other timing related problems that can occur when the Network Device Manager default engine behavior is used.</p> <p>Advanced Network Event manipulation:</p> <p>The nettrigger() function in combination with the nepending() function gives direct control of the Network Events initiated by the controller. This allows for programmatic control and monitoring of the Network Event sequence should changing the default behavior of the ScadaWorks Network Device Manager be desired. If care is used to configure ScadaBuilder's Network Destinations and subsequent Network Events by using numeric references in the names, iterating through and configuring handles through a little string manipulation (concatenating numbers to the end of Network Destination and Network Event record names as handles are configured) can facilitate looping through configurations to get the Network Event handles and storing them off. From there only the nettrigger() and nepending() need to use the Network Event handles and the Network Destination and Network Session handles can then be disregarded.</p>

<i>ST Example</i>	<pre> (* only initialize handles once *) if not init then init := TRUE; (* get the Network Session handle, may be reused for more Network Destinations*) ns := nshandle('ModbusMaster'); if ns <> 0 then (* get the Network Destination handle, may be reused for more Network Events *) nd := ndhandle(ns, 'Slave1'); if nd <> 0 then (* get the Network Event handle, *) (* may now be used for nettrigger() and nepending() functions *) netevent := nehandle(nd, 'Slave1NE1'); end_if; (* nd <> 0 *) end_if; (* ns <> 0 *) end_if; (* not init *) ... (* trigger the event *) if trigCondition and netevent <> 0 then result := nettrigger(netevent); end_if; (* monitor the progress of the Network Event *) if netevent <> 0 then ne_pending := nepending(netevent); end_if; </pre>
-------------------	--

Read Network Event State -- nepending()

<i>Type</i>	C Function
<i>Function</i>	Monitor whether a Network Event has been Triggered. This function can be used in conjunction with nettrigger() but will also detect if the Network Device Manager has triggered the Network Event from a ScadaBuilder configuration such as a Trigger.
<i>Syntax</i>	DINT nepending(DINT NetworkEventHandle);
<i>FBD</i>	
<i>Parameters</i>	nehnd - NetworkEventHandle DINT Handle from the nehandle() function. DO NOT PASS ANY OTHER PARAMETER other than a handle provided by the nehandle() or gethandles() functions.

<i>Return Value</i>	rsIt - BOOL returns a TRUE if the Network Event is currently in process. Returns false if it is not. If the Network Destination for the Network Event Handle is currently in comm fail and the Network Event has been triggered, rsIt will continue to stay TRUE with one exception. If Probing is turned off in the Network Session, then the output will be set to FALSE after retries have been exhausted. See <i>Probe Interval Disable</i> (on page 225) for more details on this functionality.
<i>Remarks</i>	This function in combination with the netrigger() function can be very helpful in resolving read before write issues and other timing related problems that can occur when using Triggered Network Events from the ScadaBuilder configuration.
<i>ST Example</i>	<pre> (* only initialize handles once *) if not init then init := TRUE; else return; end_if; (* not init *) (* get the Network Session handle, may be reused for more Network Destinations*) ns := nshandle('ModbusMaster'); if ns <> 0 then (* get the Network Destination handle, may be reused for more Network Events *) nd := ndhandle(ns, 'Slave1'); if nd <> 0 then (* get the Network Event handle, *) (* may now be used for netrigger() and nepending() functions *) netevent := nehandle(nd, 'Slave1NE1'); end_if; (* nd <> 0 *) end_if; (* ns <> 0 *) ... (* trigger the event *) if trigCondition and netevent <> 0 then result := netrigger(netevent); end_if; (* monitor the progress of the Network Event *) if netevent <> 0 then ne_pending := nepending(netevent); end_if; </pre>

File Transfer Protocol (FTP) Functions

FTP functions allow you to send and receive files to/from an FTP server over a TCP/IP connection. In order to use FTP, it must be enabled in ScadaBuilder **Node | Settings | Ethernet / Serial IP** tab.



The ISaGRAF FTP functions below are there for legacy purposes. If creating a new application that needs FTP client capability, please see ***Creating an FTP Client Interface*** (on page 373) for a more integrated feature.

These function are not supported on Pinnacle and later controllers.

Open An FTP Connection -- FtpOpen()

<i>Type</i>	C Function
<i>Function</i>	Opens an FTP client connection to a remote FTP server.
<i>Syntax</i>	ANA FtpOpen(MSG ipAddr, MSG user, MSG password);
<i>Parameters</i>	ipAddr: IP address of FTP server to connect to, in dotted decimal form user: user name for login to FTP server password: password for login to FTP server
<i>Return Value</i>	0 = success, nonzero = failure
<i>Remarks</i>	Before file transfers may be initiated by the ICL-4300 (using FtpGet and FtpSend), an FTP connection must be successfully opened with this function. This non- blocking function returns immediately. The actual opening of the connection is then handled in the background. The completion of the open connection can be determined by calling FtpCStat. Only one connection can be open at a time.
<i>ST Example</i>	(* open a connection to IP address 192.168.1.199 *) result := FtpOpen('192.168.1.199', 'jsmith', 'xj79');

Close An FTP Connection -- FtpClose()

<i>Type</i>	C Function
<i>Function</i>	Closes any open FTP client connection to a remote FTP server.
<i>Syntax</i>	ANA FtpClose();
<i>Parameters</i>	None.
<i>Return Value</i>	0 = success, nonzero = failure

<i>Remarks</i>	<p>Only one connection can be open at a time. You must close any open connection with this function before you can open a new one.</p> <p>This non-blocking function returns immediately. The actual closing of the connection is then handled in the background. The completion of the connection termination can be determined by calling FtpCStat.</p>
<i>ST Example</i>	<p>(* close any open FTP connection to remote server *)</p> <pre>result := FtpClose();</pre>

Get A File Over FTP -- FtpGet()

<i>Type</i>	C Function
<i>Function</i>	Gets a file from a remote FTP server.
<i>Syntax</i>	ANA FtpGet(MSG remote, MSG local);
<i>Parameters</i>	<p>remote: path and name of file to get from FTP server</p> <p>local: path and name of file to store locally</p>
<i>Return Value</i>	0 = success, nonzero = failure (no connection, etc.)
<i>Remarks</i>	<p>You must have established a connection to the remote FTP server (using FtpOpen) before you call this function to get a file.</p> <p>Only one file transfer can be active at any given time.</p> <p>This non-blocking function returns immediately. The actual transfer of the file happens in the background. The completion of the file transfer can be determined by calling FtpCStat.</p>
<i>ST Example</i>	<p>(* get file params.txt and store locally in p1.txt *)</p> <pre>result := FtpGet('params.txt', 'p1.txt');</pre>

Send A File Over FTP -- FtpSend()

<i>Type</i>	C Function
<i>Function</i>	Sends a file to a remote FTP server.
<i>Syntax</i>	ANA FtpSend(MSG local, MSG remote, ANA mode);
<i>Parameters</i>	<p>local: path and name of local file to send to the server</p> <p>remote: path and name of file to store remotely on server</p> <p>mode: 0 - replace any existing file 1 - append to any existing file</p>
<i>Return Value</i>	0 = success, nonzero = failure (no connection, etc.)

<i>Remarks</i>	<p>You must have established a connection to the remote FTP server (using FtpOpen) before you call this function to send a file.</p> <p>Only one file transfer can be active at any given time.</p> <p>This non-blocking function returns immediately. The actual transfer of the file happens in the background. The completion of the file transfer can be determined by calling FtpCStat.</p>
<i>ST Example</i>	<pre>(* send file data.log and store remotely as d.log, * replacing any existing file of the same name *) result := FtpSend('data.log', 'd.log', 0);</pre>

Get The FTP Client Status -- FtpCStat()

<i>Type</i>	C Function
<i>Function</i>	Reports the status of an FTP client command.
<i>Syntax</i>	ANA FtpCStat();
<i>Parameters</i>	None.
<i>Return Value</i>	<p>0 Success - command was successfully completed</p> <p>1 Failure - command failed (response error, timeout)</p> <p>4 InProgress - command is being processed</p> <p>5 Idle - no command was issued</p>
<i>Remarks</i>	This function returns the status of a command that was initiated by the client (local ICL-4300) to the remote FTP server. After issuing a command, you should repeatedly check the status and wait for it complete. If you try to initiate a new command while the current one is in progress, the new command will be rejected.
<i>ST Example</i>	<pre>(* open a connection *) result := FtpOpen('192.168.1.199', 'jsmith', 'xj79'); ... (* wait for the connection to complete *) if FtpCStat() = 0 then ... end_if;</pre>

Change FTP Transfer Type -- FtpType()

<i>Type</i>	C Function
<i>Function</i>	Sets the FTP file transfer type.
<i>Syntax</i>	ANA FtpType(ANA type);

<i>Parameters</i>	type: 0 - ASCII 1 – binary
<i>Return Value</i>	0 = success, nonzero = failure (no connection, etc.)
<i>Remarks</i>	<p>You must have established a connection to the remote FTP server (using FtpOpen) before you call this function to set the file transfer type.</p> <p>ASCII mode is typically used for text files, and binary mode is typically used for images such as executables (*.exe/*.com). Each time a connection is established the type defaults to ASCII.</p>
<i>ST Example</i>	(* set file transfer type to binary *) result := FtpType(1);

Low Level I/O Port Access Functions

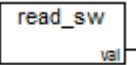
These specialized functions are provided for direct access to CPU I/O ports. Normal ISaGRAF applications will not need these functions. They are provided to allow access to specialized/custom I/O boards and hardware.



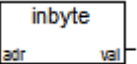
Don't use these functions unless you know what you are doing., or you may "crash" your Controller.


These functions are not supported on Pinnacle and later controllers.

Read The Eight Position DIP Switch Value -- Read_sw()

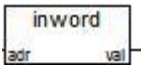

Type	C Function
Function	Read the value of the address dipswitch on controller.
Syntax	ANA Read_sw();
FBD	
Parameters	None.
Return Value	val 0-255
Remarks	Allows user to get a value from the address switch on controller. Supported controllers include the ICL Scadaflex and Etherlogic product lines.
ST Example	<pre>(* read a value from switch *) value := Read_sw(); if value > 0 then enable := TRUE; end_if;</pre>

Read One Byte From I/O Space -- InByte()

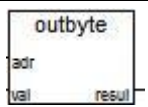
Type	C Function
Function	Read a byte value from a low-level hardware input port.
Syntax	ANA InByte(ANA adr);
FBD	
Parameters	adr: identifies the port (0-65535)
Return Value	The byte value that was read from the port.


<i>Remarks</i>	Typical users do not need this function. It provides a means of reading from custom I/O boards and hardware. Don't use this function unless you know what you are doing.
<i>ST Example</i>	(* read value from port E100 [hex] *) val := InByte(16#E100);
 <p><i>Don't use these functions unless you know what you are doing., or you may "crash" your Controller.</i></p>	

Read A 16bit Word From I/O Space -- InWord()

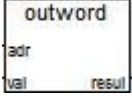

<i>Type</i>	C Function
<i>Function</i>	Read a word (2-byte) byte value from a low-level hardware input port.
<i>Syntax</i>	ANA InWord(ANA adr);
<i>FBD</i>	
<i>Parameters</i>	adr: identifies the port (0-65535)
<i>Return Value</i>	The word value that was read from the port.
<i>Remarks</i>	Typical users do not need this function. It provides a means of reading from custom I/O boards and hardware. Don't use this function unless you know what you are doing.
<i>ST Example</i>	(* read value from port E100 [hex] *) val := InByte(16#E100);
 <p><i>Don't use these functions unless you know what you are doing., or you may "crash" your Controller.</i></p>	

Write A Byte Out To I/O Space -- OutByte()

<i>Type</i>	C Function
<i>Function</i>	Write a byte value to a low-level hardware output port.
<i>Syntax</i>	ANA OutByte(ANA adr, ANA val);
<i>FBD</i>	
<i>Parameters</i>	adr: identifies the port (0-65535) val: the value to write (0-255)

<i>Return Value</i>	unused
<i>Remarks</i>	Typical users do not need this function. It provides a means of writing to custom I/O boards and hardware. Don't use this function unless you know what you are doing.
<i>ST Example</i>	(* write value of 1 to output port E000 [hex] *) result := OutByte(16#E000, 1);
 <p><i>Don't use these functions unless you know what you are doing., or you may "crash" your Controller.</i></p>	

Write A Word Out To I/O Space -- OutWord()

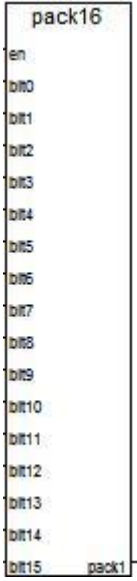
<i>Type</i>	C Function
<i>Function</i>	Write a word (2-byte) value to a low-level hardware output port.
<i>Syntax</i>	ANA OutWord(ANA adr, ANA val);
<i>FBD</i>	
<i>Parameters</i>	adr: identifies the port (0-65535) val: the value to write (0-65535)
<i>Return Value</i>	unused
<i>Remarks</i>	Typical users do not need this function. It provides a means of writing to custom I/O boards and hardware. Don't use this function unless you know what you are doing.
<i>ST Example</i>	(* write value of 1000 to output port E000 [hex] *) result := OutByte(16#E000, 1000);
 <p><i>Don't use these functions unless you know what you are doing., or you may "crash" your Controller.</i></p>	

Bit Packing and Unpacking functions

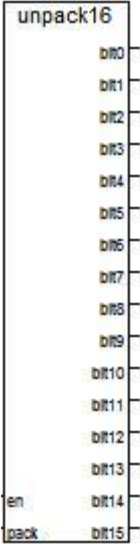
Two functions are provided so the user may pack 16 binary (boolean) values into one integer register and unpack that same register back into boolean values again. This can be used to eliminate having to setup two Network Events: one for integer registers and one for boolean registers. Once packed, the register can be placed by index where another Network Event can include it in it's block.

The Unpack16 function is a good way to decode status bits from other types of controllers including loop controllers and Variable Frequency Drive (VFD) controllers.

Pack 16 Booleans Into An Integer Register -- pack16()

<i>Type</i>	C Function
<i>Function</i>	Pack16 takes 16 boolean inputs, bit 0 through bit 15 and packs them into an integer register.
<i>Syntax</i>	ANA Pack16(BOO en, BOO bit0, BOO bit1, BOO bit2, BOO bit3, BOO bit4, BOO bit5, BOO bit6, BOO bit7, BOO bit8, BOO bit9, BOO bit10, BOO bit11, BOO bit12, BOO bit13, BOO bit14, BOO bit15);
<i>FBD</i>	
<i>Parameters</i>	Enbl: BOO enable or disable block Bit0, Bit1, Bit2, Bit3, Bit4, Bit5, Bit6, Bit7 Bit8, Bit9, Bit10, Bit11, Bit12, Bit13, Bit14, Bit15 BOO Bits to pack.
<i>Return Value</i>	pack: ANA 0 - 65535 Integer
<i>Remarks</i>	None.
<i>ST Example</i>	(* pack all alarm bits into the packBits integer register *) packedBits := pack16(alarm1, alarm2, alarm3, alarm4, alarm5, alarm6, alarm7, alarm8, alarm9, alarm10, alarm11, alarm12, alarm13, alarm14, alarm15, alarm16);

Unpack 16 Booleans From An Integer Register -- Unpack16()

Type	C Function Block
Function	Unpack16 takes an integer register and unpacks it to 16 boolean outputs bit 0 through bit 15.
Syntax	ANA (BOO en, ANA pack);
FBD	
Parameters	en: BOO enable or disable block pack: ANA 0 - 65535 Integer
Return Value	Bit0-15: BOO unpacked bits.
Remarks	To use the function block in structured text , you must declare a function block instance for each operation used.

<i>ST Example</i>	<pre>(* decode is a function block instance of unpack16 *) (* unpack all alarm bits from the packBits integer register *) result := decode(true, packedBits); (* stuff the decoded bits into boolean registers *) alarm1 := decode.bit0; alarm2 := decode.bit1; alarm3 := decode.bit2; alarm4 := decode.bit3; alarm5 := decode.bit4; alarm6 := decode.bit5; alarm7 := decode.bit6; alarm8 := decode.bit7; alarm9 := decode.bit8; alarm10 := decode.bit9; alarm11 := decode.bit10; alarm12 := decode.bit11; alarm13 := decode.bit12; alarm14 := decode.bit13; alarm15 := decode.bit14; alarm16 := decode.bit15;</pre>
-------------------	---

Instrumentation Functions

These are functions that are specific to dealing with different I/O conversions and control.

PID Closed Loop Control -- ICLPID

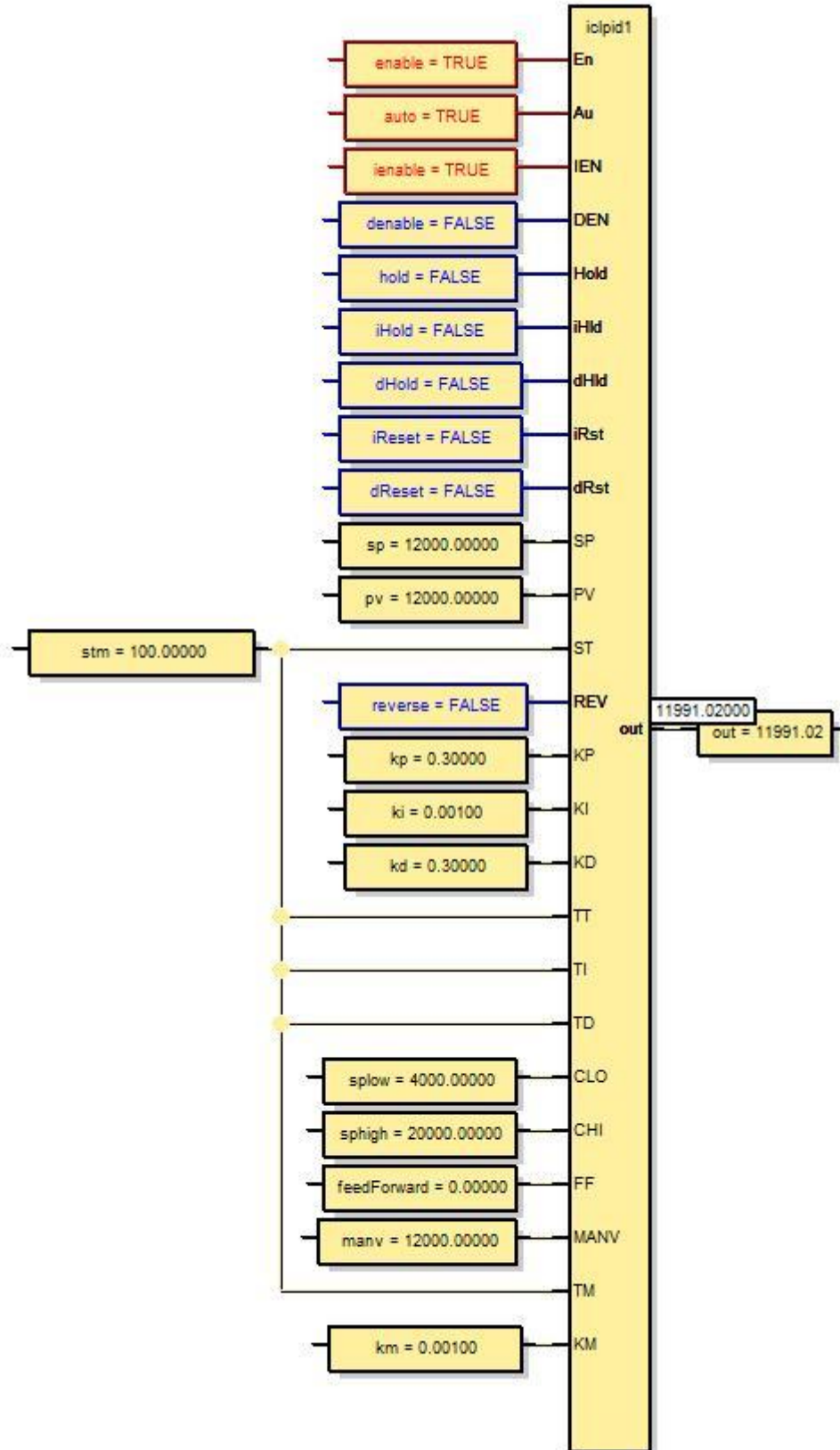
Type	C Function Block
Function	<p>Proportional, Integral, and Derivative control is used in cases where closed loop control is desired to regulate a varied load and analog or digital control loop. Such processes include variable level control, pump speed control, or temperature control. There are many other applications that can utilize a PID algorithm but all involve using a variable control range to control a process where long delays might be incurred between a process stimulus (such as an analog output) and a process reading (such as an analog input).</p> <p>The following block diagram shows an example of such process:</p> <pre> graph TD Stimulus[External Stimulus] --> Process((Variable Load Process)) Process -- "Control Output" --> Output[Output] Output --> PID[PID Algorithm] PID --> Input[Input] Input -- "Process Variable (Reading)" --> Process </pre> <p>To be the most effective, the PID algorithm must have as much linear control over the process as possible. The “gain” of the process can be adjusted at the output and input stage to vary the amount of effect the controller has on the process.</p> <p>The ICLPID also supports "bumpless" manual to automatic and automatic to manual switchover. For this to work the KM parameter must be set to a non zero value. Usually setting it to the same value as the KI parameter is sufficient.</p> <p>While the ICLPID is in Auto mode, it tracks the Integral factor to the Manual mode setpoint (ManV). When the Au (Auto) input is set to FALSE, the block uses the Manual mode configurations and integral factor to soften the transition to the ManV setpoint. While in Manual mode, the Auto mode integral factor is maintained so that the block is ready to go into Auto mode control without a large error bump. When the Au (Auto) input is set to true, the PID Auto mode configuration takes over again giving a smooth transition back to the SP target.</p>

<i>Parameters</i>	<p>En: BOOL -- Function Block Enable</p> <p>Used for Quick Ladder Implementation and when set to FALSE, completely disables all calculations (the output value is not set but remains at its last written value).</p> <p>Au: BOOL -- Auto/Manual Mode</p> <p>When set to TRUE, PID processes the error PID calculation according to the non-manual configuration.</p> <p>When set to FALSE, the ICLPID block will attempt to transfer to the ManV setpoint according to the manual configuration.</p> <p>IntEn: BOOL -- Integral Enable</p> <p>When set to FALSE, Integral Enable stops the integral value calculation but leaves the internal Integral parameter unchanged. Useful for troubleshooting tuning issues or stopping the I portion when incoming data is not valid.</p> <p>If you do not want to use integral control, then tie this input to FALSE.</p> <p>DerEn: BOOL -- Derivative Enable</p> <p>When set to FALSE, Derivative Enable stops the Derivative value calculation but leaves the internal Derivative parameter unchanged. Useful for troubleshooting tuning issues or stopping the D portion when incoming data is not valid.</p> <p>If you do not want to use derivative control, then tie this input to FALSE.</p> <p>Hold: BOOL -- ID Hold</p> <p>When set to TRUE, Hold stops all processing holds the Internal values and output value until set to FALSE again. Manual mode may be used while hold is active to override this behavior and set the output to the ManV setpoint.</p> <p>iHld: BOOL -- Hold Integral Coefficient</p> <p>When set to TRUE, Integral Hold stops the I portion of the $P + I + D + FF$ calculation but holds the Integral value. Can be used for tuning the KP parameter. Auto mode only.</p> <p>dHld: BOOL -- Hold Derivative Coefficient</p> <p>When set to TRUE, dHld stops the D portion of the $P + I + D + FF$ calculation but holds the Derivative value. Auto mode only.</p> <p>iRst: BOOL -- Reset Integral Coefficient</p> <p>When set to TRUE, Sets the Integral portion of the $P + I + D + FF$ calculation to zero. When set back to FALSE the Integral calculation starts from 0.0.</p> <p>dRst: BOOL -- Reset Derivate Coefficient</p> <p>When set to TRUE, Sets the Derivative portion of the $P + I + D + FF$ calculation to zero. What set back to FALSE the Derivative calculation starts from 0.0.</p>
-------------------	---

<p><i>Parameters continued...</i></p>	<p>SP: REAL -- Process Setpoint</p> <p>Target which the PID calculates the current error.</p> <p>In Reverse acting mode the error is calculated as $PV - SP$. In normal mode (non-Reverse acting) the error is calculated as $SP - PV$.</p> <p>PV: REAL -- Process Variable</p> <p>Feedback from which the PID calculates the current error.</p> <p>In Reverse acting mode the error is calculated as $PV - SP$. In normal mode (non-Reverse acting) the error is calculated as $SP - PV$.</p> <p>ST: REAL -- Sample Time (Delta) in milliseconds</p> <p>This parameter is used in both Auto and Manual modes as the primary Delta Time for both Integral and Derivative calculations. It should be set to a value greater than the ISaGRAF scan time of the application. If the value is less than the scan time, then tuning the KI and KD parameters can be difficult due to variation in program scan time.</p> <p>Recommended value to start with for most programs is 100.0 unless the programs scan time is longer.</p> <p>REV: BOOL -- Reverse Acting PID</p> <p>When set to TRUE reverses the direction of the error calculated by the PID and reverses the correcting action.</p> <p>In Reverse acting mode the error is calculated as $PV - SP$. In normal mode (non-Reverse acting) the error is calculated as $SP - PV$.</p> <p>KP: REAL -- Proportional Coefficient Setpoint</p> <p>The PID calculates the error and applies this parameter as a multiplier error every program scan and applies it to the P portion of the $P + I + D + FF$ output. This output is then clamped according to the the CHI and CLO parameters after the output is calculated.</p> <p>Recommend value for a 1/1 process is about 0.3</p> <p>KI: REAL -- Integral Coefficient Setpoint</p> <p>The PID calculates the error then integrates the I portion and then applies this multiplier to the $P + I + D + FF$ output. This output is then clamped according to the the CHI and CLO parameters after the output is calculated. If the output is clamped (out of range then set to the limit), then the I portion of the PID is no longer calculated. This is an anti-windup feature of the ICLPID block.</p> <p>Recommend value for a 1/1 process is about 0.001</p> <p>KD: REAL -- Derivative Coefficient Setpoint</p> <p>The PID calculates the error then the D portion of the PID. It then applies this multiplier to the $P + I + D + FF$ output. This output is then clamped according to the the CHI and CLO parameters after the output is calculated. If the output is clamped (out of range then set to the limit), then the D portion of the PID is no longer calculated. This is an anti-spike feature of the ICLPID block.</p> <p>Recommend value for a 1/1 process is about 0.1</p>
---------------------------------------	---

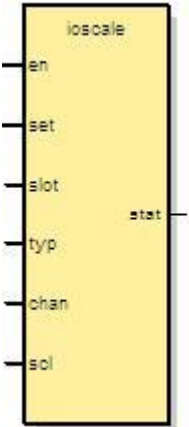
<i>Parameters Continued</i>	<p>TT: REAL-- Tracking Time</p> <p>Used to transfer between Manual and Auto modes and is used in both Auto Mode and Manual Mode Integral calculations. It is recommended setting this parameter to the same as the ST parameter and greater than the program scan cycle.</p> <p>TI: REAL -- Tracking Time for Integral Calculation</p> <p>Used in Auto mode and from Manual to Auto mode transitions. It is recommended setting this parameter to the same as the ST parameter and greater than the program scan cycle.</p> <p>TD: REAL -- Tracking Derivative Time</p> <p>Used to transfer used in Auto mode only. It is recommended setting this parameter to the same as the ST parameter and greater than the program scan cycle.</p> <p>CLO: REAL -- Output Clamp Low Setpoint -- Used for Auto and Manual modes. Provides Integral anti-windup, Proportional/Derivative and Feed Forward out-of-range protection.</p> <p>Clamping is done on the output value only expressed as:</p> $\text{out} = \text{clamp}(\text{outRaw}, \text{CLO}, \text{CHI})$ <p>CHI REAL -- Output Clamp High Setpoint -- Used for Auto and Manual modes. Provides Integral anti-windup, Proportional/Derivative and Feed Forward out-of-range protection.</p> <p>Clamping is done on the output value only expressed as:</p> $\text{out} = \text{clamp}(\text{outRaw}, \text{CLO}, \text{CHI})$ <p>FF: REAL - Feed Forward</p> <p>Added to output before clamping ($P + I + D + FF$)</p> <p>ManV: REAL -- Manual Mode Control Value</p> <p>When AU(to) is set to false, this is the target setpoint that the output will move to using the Manual mode parameters.</p> <p>TM: REAL -- Track Time Manual Mode</p> <p>Used for Manual mode Integral calculations to transfer from Auto to Manual modes. It is recommended setting this parameter to the same as the ST parameter and greater than the program scan cycle.</p> <p>KM: REAL - Manual Mode Gain</p> <p>Used to transfer from Manual to Auto modes. This is the proportion of Integral applied to the Manual Mode I coefficient.</p> <p>It is recommend to set this value to the same as the KI value to start out with.</p> <p>Transitions from Auto to Manual mode setpoint can be extreme. Lower this value to get a slower transition, raise it for a faster transition.</p>
<i>Return Value</i>	<p>OUT: REAL PID process correction output.</p> <p>Value to be passed to process being controlled by the ICLPID block. Typically this would connect to an analog output of a controller.</p>

FBD Sample



Remarks	<p>Shown here is a live running ICLPID block with the recommended default configuration and connections.</p> <p>This a 1/1 (input to output) operation with no scaling between the two. If scaling is required for the process, it is usually be done on the output of the ICLPID block.</p> <p>ISaGRAF 5 Controllers (Pinnacle and later) only.</p> <p>To use the function block in structured text , you must declare a function block instance for each operation used.</p>
---------	--

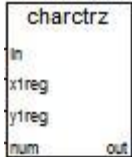
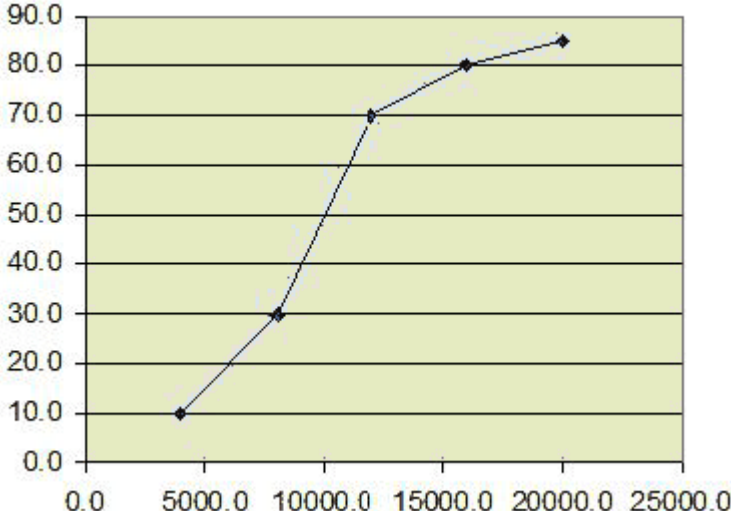
Apply Scaling Record To Analog Channel -- ioscale()

Type	C Function Block
Function	Apply a different ScadaBuilder Scaling Record to an analog I/O channel.
Syntax	DINT ioscale(BOOL en, BOOL set, DINT slot, STRING typ, DINT chan, STRING scl);
FBD	

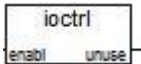
<i>Parameters</i>	<p>en: BOOL set to true to run function block. This is for LD language compatibility.</p> <p>set: BOOL set to true to apply the current parameters below. Must be toggled FALSE for one scan then TRUE again to reapply and changed parameters after the initial application.</p> <p>slot: DINT slot number where channel is located.</p> <p>0 = CPU (top slot).</p> <p>1 = First option card (middle card).</p> <p>2 = Second options card (bottom card toward bottom of case)</p> <p>typ: STRING type of I/O scaling record is applied to (where the analog I/O is located in the ScadaBuilder configuration:</p> <p>'UI' = Universal Input section.</p> <p>'AI' = Analog Input section (includes Internal Temperature and Power Monitoring.</p> <p>'AO' = Analog Output section.</p> <p>chan: DINT channel number physically on the card counted from the first one to the last of that type as they are shown in by channel in the ScadaBuilder I/O map.</p> <p>scl: STRING name of Scaling Record configured in ScadaBuilder. This name is case sensitive.</p>
<i>Return Value</i>	<p>stat: DINT status output after operation. Any negative number indicates an error.</p> <p>0 = Scaling Record applied successfully.</p> <p>-1 = Invalid Slot number.</p> <p>-2 = Invalid Type.</p> <p>-3 = Invalid channel number.</p> <p>-4 = Invalid Scaling Record string.</p> <p>-5 = Scaling Record does not exist.</p> <p>-6 = Error applying Scaling Record.</p>
<i>Remarks</i>	To use the function block in structured text , you must declare a function block instance for each operation used.
<i>ST Example</i>	<p>(* Apply the bipolar millivolt scaling to the CPU slot UI channel 6 *)</p> <p>result := Totalize(TRUE, togglebit, 0, 'UI', 6, '-250 to 250mV',);</p> <p>(* reset toggle bit for next scaling application *)</p> <p>togglebit := false;</p>

Characterize A Non-Linear Instrumentation Curve -- charctrz()

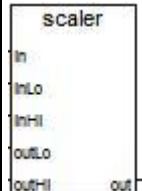
<i>Type</i>	C Function
-------------	------------

Function	Provides 'num' segments that can characterize the input signal. Segments are configured by entering the X1...Xnum and Y1...Ynum register points. All Xn+1 register points must be greater than the Xn register points.																								
Syntax	REAL charctrz(REAL in, ANA x1Reg, ANA y1Reg, ANA num);																								
FBD																									
Parameters	<p>in: REAL Input X signal.</p> <p>x1reg: ANA First register (REAL) address by index for X coordinate segments.</p> <p>y1reg: ANA First register (REAL) address by index for Y coordinate segments.</p> <p>num: ANA Number of X/Y segment register pairs (the segment registers that characterize each coordinate must be REAL registers with sequential addresses).</p>																								
Return Value	out: REAL Output Y signal																								
Remarks	<p>Each X and Y coordinate must greater than the previous coordinate. Full linearization will be calculated between coordinate intersections (see below).</p> <p>If the following system has these values in its registers</p>																								
ST Example	<table><tr><th>X Register Index</th><th>Value</th><th>Y Register Index</th><th>Value</th></tr><tr><td>4005</td><td>85.0</td><td>4011</td><td>4000.0</td></tr><tr><td>4004</td><td>80.0</td><td>4012</td><td>8000.0</td></tr><tr><td>4003</td><td>70.0</td><td>4013</td><td>12000.0</td></tr><tr><td>4002</td><td>30.0</td><td>4014</td><td>16000.0</td></tr><tr><td>4001</td><td>10.0</td><td>4015</td><td>20000.0</td></tr></table> <p>Calling the charctrz function like this:</p> <p>output := charctrz (analogInput, 4011, 4005, 5);</p> <p>will yield a curve that linearizes the above chart of numbers.</p> 	X Register Index	Value	Y Register Index	Value	4005	85.0	4011	4000.0	4004	80.0	4012	8000.0	4003	70.0	4013	12000.0	4002	30.0	4014	16000.0	4001	10.0	4015	20000.0
X Register Index	Value	Y Register Index	Value																						
4005	85.0	4011	4000.0																						
4004	80.0	4012	8000.0																						
4003	70.0	4013	12000.0																						
4002	30.0	4014	16000.0																						
4001	10.0	4015	20000.0																						

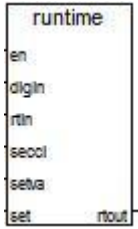
Enable Or Disable The Controller I/O Scan -- IOCtrl()

Type	C Function
Function	Enables or disables I/O scanning.
Syntax	ANA IOCtrl(BOO enable);
FBD	
Parameters	enable: BOO Set TRUE to enable I/O scanning. Set FALSE to disable I/O scanning.
Return Value	unused
Remarks	This function is used to enable or disable I/O scanning. If I/O scanning is disabled, then no input/output updates will take place. If you would like to start your program with I/O scanning disabled see Node Settings Advanced (see "Node Settings - Advanced Tab" on page 47) tab for more details.
ST Example	(* disable I/O scanning *) result := IOCtrl(FALSE);


Scale A Linear Analog Device -- Scaler()

Type	C Function
Function	Scale a REAL type input register to a REAL type output register.
Syntax	REAL Scaler(REAL in, REAL inLo, REAL inHi, REAL outLo, REAL outHi);
FBD	
Parameters	in: input register value inLo: input low scale range inHi: input high scale range outLo: output low scale range outHi: output high scale range
Return Value	out: The scaled output register value.
Remarks	Scaling can also be done in the I/O Scaling section of Scadabuilder.
ST Example	(* scale the input value from the instrument *) scaledValue := scaler(REAL(aiTankLevel), 4000.0, 20000.0, 0.0, 34.62);

Totalize The Time A Boolean Is True -- Runtime()


<i>Type</i>	C Function Block
<i>Function</i>	Runtime totalizes the time a digital input or other boolean source stays true.
<i>Syntax</i>	ANA Runtime(BOO en, BOO digin, ANA rtin, ANA secclk, ANA setval, BOO set);
<i>FBD</i>	
<i>Parameters</i>	<p>en: BOO When false will inhibit all functionality.</p> <p>digin: BOO Boolean signal to be tracked while true</p> <p>rtin: ANA Analog value that is in the time units specified. This variable must be the same as connected to rtout and should also be retained.</p> <p>secclk: ANA Multiplier where the number of seconds times this will be the accumulated value of the output. For example, a value of six will totalize in .1 minutes. A value of 60 will totalize in minutes. A value of 0 or less defaults to 1 second.</p> <p>setval: ANA Value to be written to the rtout output when the set input is set to true. Only 0 or positive values are allowed.</p> <p>set: BOO Set to true to write the setval value to the rtout output.</p>
<i>Return Value</i>	rtout:: ANA Totalized time the digin input has remained true. The variable tied to this output should also be tied to rtin. It should also be set to retained.
<i>Remarks</i>	To use the function block in structured text , you must declare a function block instance for each operation used.
<i>ST Example</i>	<pre>(* runtm is a function block instance of Runtime *) result := runtm (TRUE, PumpRunContact, pumpRunTime, 60, 0, FALSE); pumpRunTime := runtm.rtout;</pre>

Periodically Totalize An Analog Value -- Totalize()

<i>Type</i>	C Function Block
<i>Function</i>	Totalizes an analog value over time.
<i>Syntax</i>	REAL Totalize(BOO en, REAL in, REAL totin, ANA secclk, REAL setval, BOO set);
<i>FBD</i>	

<i>Parameters</i>	<p>en: BOO When false will inhibit all functionality.</p> <p>in: REAL signal to be totalized (should be in some periodic units).</p> <p>totin: REAL totalized value that is in the time units specified. This variable must be the same as connected to totout and should also be retained. It is read once by the block at startup.</p> <p>secclk: ANA Number of 1 second sampling periods per totalization. For example, a value of six will totalize in 6 seconds. A value of 60 will totalize in minutes. A value of 0 or less defaults to 1 second.</p> <p>setval: REAL Value to be written to the totout output when the set input is set to true. Only 0 or positive values are allowed.</p> <p>set: BOO Set to true to write the setval value to the totout output.</p>
<i>Return Value</i>	totout: REAL Totalized value of the in input. The variable tied to this output should also be tied to totin. It should also be set to retained.
<i>Remarks</i>	To use the function block in structured text , you must declare a function block instance for each operation used.
<i>ST Example</i>	<pre>(* totize is a function block instance of Totalize *) result := totize (TRUE, flowGpm, averageTotalGallons, 60, 0, FALSE); averageTotalGallons := totize.rtout;</pre>

Track And Hold An Analog Control Value -- trackhld()

<i>Type</i>	C Function Block
<i>Function</i>	Holds an initial value transferred to output on first scan. Tracks the TrackVariable when TrackCommand is TRUE and holds the last output value when FALSE.
<i>Syntax</i>	REAL trackhld(REAL init, REAL tv, BOO tc);
<i>FBD</i>	
<i>Parameters</i>	<p>INIT: REAL Initial value to transfer to Output.</p> <p>TV: REAL (TrackVariable) Input signal to track.</p> <p>TC: BOO (Track command) When TRUE, Output tracks the TrackVariable. When FALSE, Output stays the same as the last Output value.</p>
<i>Return Value</i>	OUT: REAL Output signal.
<i>Remarks</i>	To use the function block in structured text , you must declare a function block instance for each operation used.
<i>ST Example</i>	<pre>(*track is a function block instance of trackhld() *) result := track(initValue, controlSetpoint, not(holdOutput)); controlOutput := track.out;</pre>

Limit Rise And Fall Rate Of An Analog Value -- ratelim()

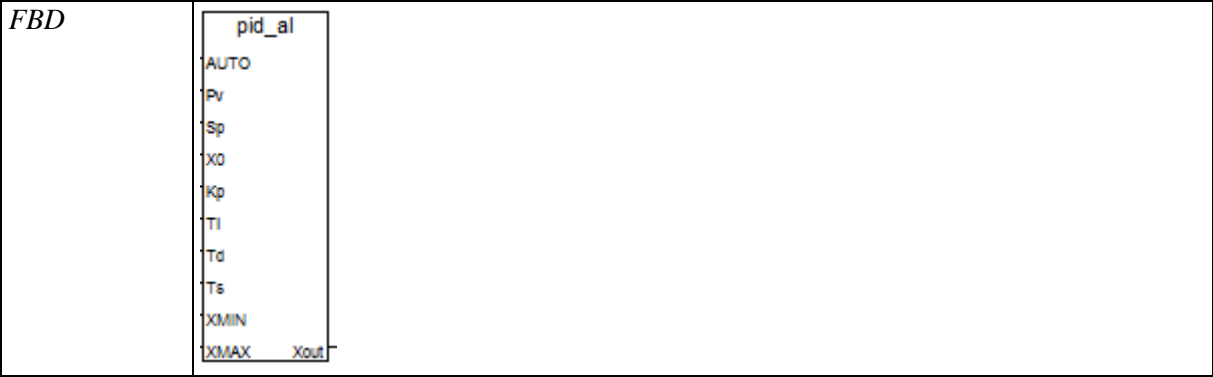
<i>Type</i>	C Function Block
<i>Function</i>	Limits the rate of change for an input signal.
<i>Syntax</i>	ANA Ratelim(REAL in, REAL up, REAL down, BOO enb);
<i>FBD</i>	
<i>Parameters</i>	<p>in: REAL (Input) Real value on which to limit the rate variation.</p> <p>up: REAL (UpRate) The upper limit rate, in units/minute.</p> <p>down: REAL (DownRate) The lower limit rate, in units/minute.</p> <p>enb: BOO (Enable) TRUE enables rate limitation action.</p>
<i>Return Value</i>	<p>out: REAL (Output) When Enable is FALSE, Output equals Input. When Enable is TRUE, Output rate is limited by UpRate or DownRate.</p> <p>rl: BOOL (RisingLimit) TRUE when block limits a rising Input.</p> <p>fl: BOOL (FallingLimit) TRUE when block limits a falling Input.</p>
<i>Remarks</i>	<p>When enb is TRUE:</p> <p>When the Input signal changes faster than the UpRate limit, the RisingLimit is TRUE and Output changes at the UpRate rate. When the Input signal changes slower than the DownRate limit, the FallingLimit is TRUE and Output changes at the DownRate rate. When the Input signal changes at a rate between UpRate and DownRate, Output tracks Input.</p> <p>When enb is FALSE:</p> <p>The Output tracks the Input.</p> <p>To use the function block in structured text , you must declare a function block instance for each operation used.</p>
<i>ST Example</i>	<pre>(* limiter is a function block instance of ratelim *) (* limit rise to 100 degrees per minute *) (* limit fall to 50 degrees per minute *) result := limiter(rampControl, 100.0, 50.0, rampLimiter) (* assign to the core control analog output *) rampOutput := limiter.out; (* record whether we are limiting or not *) limitRisingFlag := limiter.rl; limitFallingFlag := limiter.fl;</pre>

PID Closed Loop Control -- PID_AL()

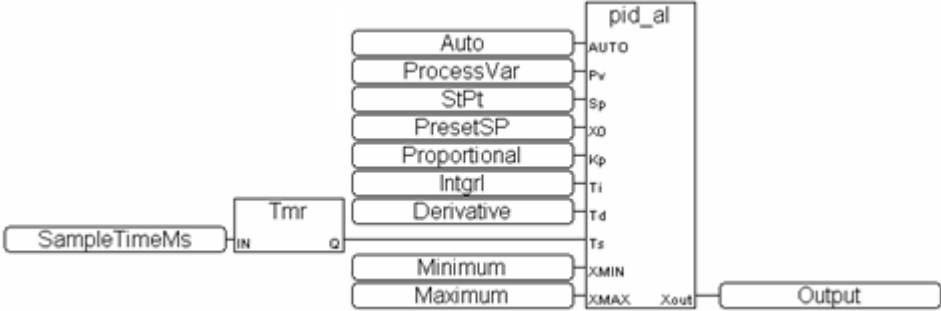
!

This function is deprecated in favor of the *PID Closed Loop Control -- ICLPID* (on page 575) function block.

Type	C Function Block
Function	<p>Proportional, Integral, and Derivative control is used in case where closed loop control is desired to regulate a varied load, analog or digital control loop. Such processes include variable level control, pump speed control, or temperature control. There are many other applications that can utilize a PID algorithm but all involve using a variable control range to control a process where long delays might be incurred between a process stimulus (such as an analog output) and a process reading (such as an analog input).</p> <p>The following block diagram shows an example of such process:</p> <pre>graph LR PV[Process Variable Reading] --> Input[Input] Input --> PID[PID Algorithm] PID --> Output[Output] Output --> CO[Control Output] CO --> PV ESP[External Stimulus] --> VLP((Variable Load Process)) VLP --> PV</pre> <p>To be the most effective, the PID algorithm must have as much linear control over the process as possible. The “gain” of the process can be adjusted at the output and input stage to vary the amount of effect the controller has on the process.</p>
Syntax	REAL pid_al(BOO auto, REAL pv, REAL sp, REAL xo, REAL kp, REAL ti, REAL td, TIMER ts, REAL xmin, REAL xmax);



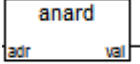
<i>Parameters</i>	<p>AUTO: BOO Automatic Control</p> <p>This Boolean is set to TRUE for automatic control and FALSE for manual control. Manual control causes the block to reflect the XO input to the OUT return value. The value is still limited by the XMIN and XMAX parameters. Setting this value to FALSE and TRUE again also resets the integrator values in the block causing the control loop to begin again.</p> <p>PV: REAL Process Variable</p> <p>The process variable is the feedback or reading input to the PID_AL block. The PID_AL block calculates the current error every program scan based on this value.</p> <p>SP: REAL Setpoint</p> <p>The setpoint is the desired target of the PID_AL block. This is the value the PID_AL block is trying to achieve at the process variable input. The PID_AL block calculates the current error every program scan based on this value.</p> <p>KP: REAL Proportional Factor</p> <p>The proportional factor essentially adjusts the "live" gain of the direct error calculation. The result is then summed with the integral and derivative values calculated at the TS sample intervals. The proportional factor itself is calculated and applied to the output every scan.</p> <p>TI: REAL Integral Factor</p> <p>The integral factor essentially adjusts the effect of calculated integral from the last TS sample period. TS and TI are inversely interactive. The longer the TS, the less the effect of the TI factor. Since the integral is essentially the level output of the block, it acts like a sample and hold using TS as the sample period. The Integrated error is then divided by the TI factor and summed with the proportional and derivative terms.</p> <p>TD: REAL Derivative Factor</p> <p>The derivative factor essentially adjusts the effect of calculated derivative from the last TS sample period. The derivative factor acts like a dampening affect on the error thus correcting large spikes in error that might be introduced as a result of the proportional error calculation. TS and TD are interactive, the longer the TS the greater the more effect the TD factor has.</p> <p>TS: TIMER Integral Time</p> <p>The integral time parameter adjusts the sample time of the integrator itself in milliseconds. It is entered as a timer value. The PID_AL recalculates integral every time the TS period elapses. TS and the TI parameter are proportionately interactive, the greater the TS, the greater the effect of TI. If the TS parameter is set to less than the scan time of the program, then the scan time of the program is used as the sample time.</p> <p>XMIN: REAL Minimum Output Limit</p> <p>XMAX: REAL Maximum Output Limit</p> <p>The XMIN and XMAX parameters work together to limit the output of the PID_AL block and prevent integral wind up within the block. If the calculated output of the PID_AL block is outside these limits, the block will a) revert back to the old integral and derivative values and b) limit the output. The XMIN and XMAX parameters also limit the output when the PID_AL block is in manual mode.</p> <p>XO: REAL Manual Mode Setpoint</p> <p>The manual mode setpoint allows another process to control the output of the PID_AL block. When the AUTO input is FALSE, the XO is reflected at the output directly. The limits of XMIN and XMAX still apply however thus limiting the output.</p>
-------------------	--

Return Value	<p>OUT: REAL Output</p> <p>Value to be passed to process being controlled by the PID_AL block. Typically this would connect to an analog output of a controller.</p>
Remarks	<p>To use the function block in structured text , you must declare a function block instance for each operation used.</p> <div><div>!</div><div><p>This function is deprecated in favor of the PID Closed Loop Control -- ICLPID (on page 575) function block.</p></div></div>
FBD Example	<div><p>The diagram shows an FBD for the PID_AL block. On the left, a 'SampleTimeMs' block is connected to the 'IN' input of a 'Tmr' block. The 'Q' output of the 'Tmr' block is connected to the 'Auto' input of the 'pid_al' block. The 'pid_al' block has several inputs: 'ProcessVar' (Pv), 'StPt' (Sp), 'PresetSP' (X0), 'Proportional' (Kp), 'Intgrl' (Ti), 'Derivative' (Td), 'Ts', 'XMIN', and 'XMAX'. The 'Xout' output of the 'pid_al' block is connected to an 'Output' block.</p></div> <p>In this example, the user has setup every setpoint needed so that when the process is running, the system will use the PID control parameters that the user specified. It is recommended to use retained variables for the setpoint inputs so they will be non-volatile through a controller reset or power cycle.</p> <p>In this case where the Run input goes false, the user can chose a setpoint or fill in a value of zero to close the valve. This will also reset the integrator of the block to be ready when the Auto input goes TRUE again.</p>

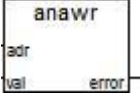
Variable Access Functions

Provides a means of accessing ISaGRAF variables using their "network address" (or register index), instead of by name. The network address is typically used in communication protocols, such as Modbus, to identify variables/registers.

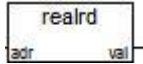
Read An Integer Register By Index -- AnaRd()

<i>Type</i>	C Function
<i>Function</i>	Read an Analog variable by its network address, as defined in the variable dictionary of the application.
<i>Syntax</i>	ANA AnaRd(ANA adr);
<i>FBD</i>	
<i>Parameters</i>	adr: ANA identifies the variable
<i>Return Value</i>	val: ANA the value to read
<i>Remarks</i>	none
<i>ST Example</i>	(* read Analog variable 17 *) value := AnaRd(17);

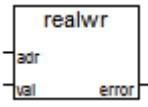
Write To An Integer Register By Index -- AnaWr()

<i>Type</i>	C Function
<i>Function</i>	Write an Analog variable by its network address, as defined in the variable dictionary of the application.
<i>Syntax</i>	ANA AnaWr(ANA adr, ANA val);
<i>FBD</i>	
<i>Parameters</i>	adr: ANA identifies the variable val: ANA the value to write
<i>Return Value</i>	unused
<i>Remarks</i>	none
<i>ST Example</i>	(* write value of 123 to Analog variable 17 *) result := AnaWr(17, 123);


Read A Real Register By Index -- RealRd()

<i>Type</i>	C Function
<i>Function</i>	Read a Real variable by its network address, as defined in the variable dictionary of the application.
<i>Syntax</i>	REAL RealRd(ANA adr);
<i>FBD</i>	
<i>Parameters</i>	adr: identifies the variable
<i>Return Value</i>	val: REAL the value to write
<i>Remarks</i>	none
<i>ST Example</i>	(* read Real variable 98 *) value := RealRd(98);

Write To A Real Register By Index -- RealWr()

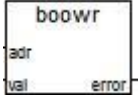
<i>Type</i>	C Function
<i>Function</i>	Write a Real variable by its network address, as defined in the variable dictionary of the application.
<i>Syntax</i>	ANA RealWr(ANA adr, REAL val);
<i>FBD</i>	
<i>Parameters</i>	adr: ANA identifies the variable val: REAL the value to write
<i>Return Value</i>	unused
<i>Remarks</i>	none
<i>ST Example</i>	(* write value of 567.8 to Real variable 35 *) result := RealWr(35, 567.8);

Read A Boolean Register By Index -- BooRd()


<i>Type</i>	C Function
<i>Function</i>	Read a Boolean variable by its network address, as defined in the variable dictionary of the application.
<i>Syntax</i>	BOO BooRd(ANA adr);
	

<i>Parameters</i>	adr: ANA identifies the variable
<i>Return Value</i>	the value of the variable, false if not found
<i>Remarks</i>	none
<i>ST Example</i>	(* read Boolean variable 100 *) value := BooRd(100);

Write To A Boolean Register By Index -- BooWr()

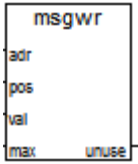
<i>Type</i>	C Function
<i>Function</i>	Write a Boolean variable by its network address, as defined in the variable dictionary of the application.
<i>Syntax</i>	ANA BooWr(ANA adr, BOO val);
<i>FBD</i>	
<i>Parameters</i>	adr: ANA identifies the variable val: BOO the value to write
<i>Return Value</i>	Unused
<i>Remarks</i>	None
<i>ST Example</i>	(* write value of true to Boolean variable 159 *) result := BooWr(159, true);

Read A Message Register By Index -- MsgRd()

<i>Type</i>	C Function
<i>Function</i>	Read a message variable by its network address, as defined in the variable dictionary of the application.
<i>Syntax</i>	MSG MsgRd(ANA adr, ANA pos, ANA max);
<i>FBD</i>	
<i>Parameters</i>	adr: ANA identifies the variable pos: ANA starting read index position within the message variable (1...255) max: ANA the maximum number of characters to read from the message variable
<i>Return Value</i>	msg: MSG the value of the variable, empty message variable if not found
<i>Remarks</i>	none

<i>ST Example</i>	<pre>(* read message variable 100, starting with 10th character, getting 15 characters maximum: *) value := MsgRd(100, 10, 15);</pre>
-------------------	--

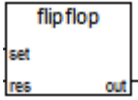
Write To A Message Register By Index -- MsgWr()

<i>Type</i>	C Function
<i>Function</i>	Write a message variable by its network address, as defined in the variable dictionary of the application.
<i>Syntax</i>	ANA MsgWr(ANA adr, ANA pos, MSG val, ANA max);
<i>FBD</i>	
<i>Parameters</i>	<p>adr: ANA identifies the variable</p> <p>pos: ANA starting write index position within the message variable (1...255)</p> <p>val: MSG the value to write</p> <p>max: ANA the maximum number of characters to write to the message variable</p>
<i>Return Value</i>	unused
<i>Remarks</i>	none
<i>ST Example</i>	<pre>(* write 'abcde' to message variable 74 *) result := MsgWr(74, 1, 'abcdefghij', 5);</pre>

Logical Functions

Logic functions are utility oriented to make things easier to program.

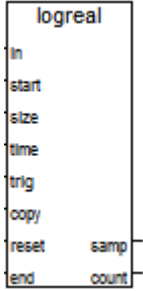
Flip Flop Latch Function -- flipflop()

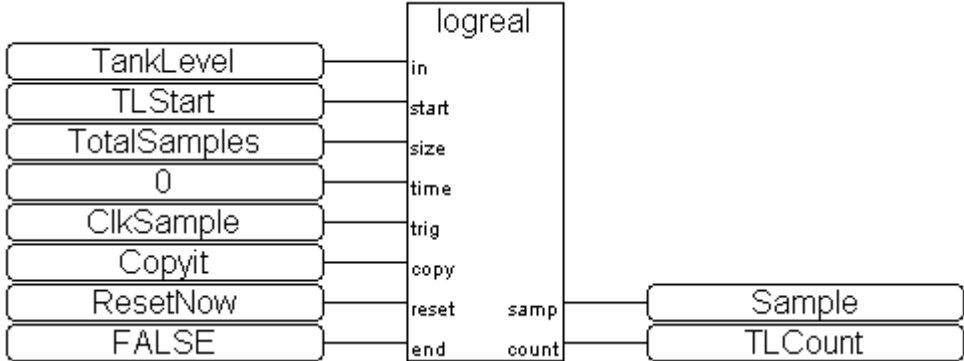
<i>Type</i>	C Function Block
<i>Function</i>	<p>Provides a Flip-Flop function as detailed in the truth table below:</p> <pre> R S LO OUT 1 X X 0 0 / 1 0 0 / 0 1 0 0 0 0 0 0 1 1 R = Reset input S = Set input LO = last output (internal value) O = output X = any state / = rising edge </pre>
<i>Syntax</i>	BOO flipflop(BOO set, BOO res);
<i>FBD</i>	
<i>Return Value</i>	out: BOO output signal.
<i>Remarks</i>	<p>Application note: good for use as a pump alternator.</p> <p>To use the function block in structured text , you must declare a function block instance for each operation used.</p>
<i>ST Example</i>	<pre> (*alternate is a function block instance of flipflop *) (*every time triggerChange goes from TRUE to FALSE and back to TRUE again *) (* set the alternate flag to the opposite state *) alternate(triggerChange, FALSE); alternateFlag := alternate.out; </pre>

Logging Functions

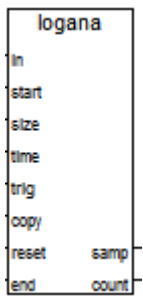
These functions are utilized to make the ISaGRAF dictionary act like a FIFO (First In First Out) buffer array. The typical application is to make an on-board trend buffer that can be accessed by a HMI (Human Machine Interface) panel and displayed as a graphical trend.

Use The Real Register Dictionary As A FIFO Log Or Trend -- Logreal()

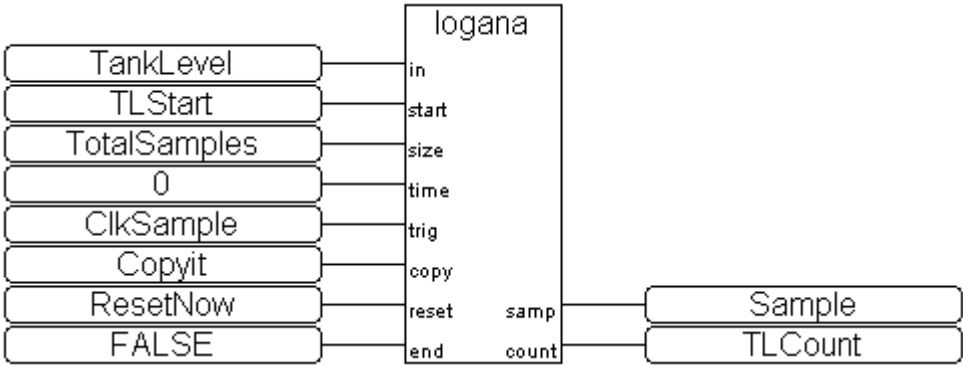
Type	C Function Block
Function	<p>This function samples a real input variable and stores the values in memory up to the number specified by the SIZE input. If the number of samples exceeds the allocated size then the most recent samples will be retained. Samples can be taken based at a specified time period (TIME) or by asserting the input trigger (TRIG).</p> <p>The samples can be copied to a block of variables (or registers) by setting the COPY input. This will cause the current number of samples indicated by the COUNT output to be copied to the block of variables starting at address specified by the START input. The END input will control the direction that the samples are copied to the block of variables.</p> <p>Setting the RESET line will cause the block of variables to be cleared and COUNT output to be set to zero.</p>
Syntax	ANA logreal(REAL in, ANA start, ANA size, ANA time, BOO trig, BOO copy, BOO reset, BOO end);
FBD	
Parameters	<p>in: REAL variable that is sampled.</p> <p>start: ANA Starting address of variable where samples are copied to.</p> <p>size: ANA Number of samples to allocate memory for and copy. This input is only read once when the program is initialized.</p> <p>time: ANA The period in milli-seconds to sample the input. Set to zero to disable sampling at a specified period.</p> <p>trig: BOO Setting to TRUE will cause the input to be sampled on the next scan.</p> <p>copy: BOO Setting to TRUE will cause the current number of samples (COUNT) to be copied to the block of variables starting at the address specified by the START input.</p> <p>reset: BOO Setting to TRUE will cause the block of variables used to copy the samples to be zeroed along with the COUNT output.</p> <p>end: BOO Specifies the direction in which samples are copied to the block of variables. A TRUE value will cause the most recent values to be copied to the end of the block. A FALSE value will cause the most recent values to be copied to the front.</p>

Return Value	<p>samp: BOO When indicates that a sample was taken on the current scan.</p> <p>count: ANA Indicates the current number of sample in the memory block. This is the number of samples that will be copied to the block of variables starting at the specified START address when the COPY input is set to TRUE.</p>
Remarks	To use the function block in structured text , you must declare a function block instance for each operation used.
FBD Example	 <p>The diagram shows an FBD example for the 'logreal' function block. On the left, there is a vertical stack of eight input boxes: 'TankLevel', 'TLStart', 'TotalSamples', '0', 'ClkSample', 'Copyit', 'ResetNow', and 'FALSE'. Each box is connected by a horizontal line to a corresponding input port on the right side of the 'logreal' block. The ports are labeled 'in', 'start', 'size', 'time', 'trig', 'copy', 'reset', and 'end' from top to bottom. From the right side of the 'logreal' block, two output lines emerge. The top line is labeled 'samp' and connects to a box labeled 'Sample'. The bottom line is labeled 'count' and connects to a box labeled 'TLCount'.</p>

Use The Integer Register Dictionary As A FIFO Log Or Trend -- Logana()

Type	C Function Block
Function	<p>This function samples a real input variable and stores the values in memory up to the number specified by the SIZE input. If the number of samples exceeds the allocated size then the most recent samples will be retained. Samples can be taken based at a specified time period (TIME) or by asserting the input trigger (TRIG).</p> <p>The samples can be copied to a block of variables (or registers) by setting the COPY input. This will cause the current number of samples indicated by the COUNT output to be copied to the block of variables starting at address specified by the START input. The END input will control the direction that the samples are copied to the block of variables.</p> <p>Setting the RESET line will cause the block of variables to be cleared and COUNT output to be set to zero.</p>
Syntax	ANA logana(ANA in, ANA start, ANA size, ANA time, BOO trig, BOO copy, BOO reset, BOO end);
FBD	
Parameters	<p>in: ANA Integer variable that is sampled.</p> <p>start: ANA Starting address of variable where samples are copied to.</p> <p>size: ANA Number of samples to allocate memory for and copy. This input is only read once when the program is initialized.</p> <p>time: ANA The period in milli-seconds to sample the input. Set to zero to disable sampling at a specified period.</p> <p>trig: BOO Setting to TRUE will cause the input to be sampled on the next scan.</p> <p>copy: BOO Setting to TRUE will cause the current number of samples (COUNT) to be copied to the block of variables starting at the address specified by the START input.</p> <p>reset: BOO Setting to TRUE will cause the block of variables used to copy the samples to be zeroed along with the COUNT output.</p> <p>end: BOO Specifies the direction in which samples are copied to the block of variables. A TRUE value will cause the most recent values to be copied to the end of the block. A FALSE value will cause the most recent values to be copied to the front.</p>
Return Value	<p>samp: BOO When indicates that a sample was taken on the current scan.</p> <p>count: ANA Indicates the current number of sample in the memory block. This is the number of samples that will be copied to the block of variables starting at the specified START address when the COPY input is set to TRUE.</p>
Remarks	To use the function block in structured text , you must declare a function block instance for each operation used.

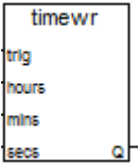
FBD Example



Real Time Clock (RTC) Functions

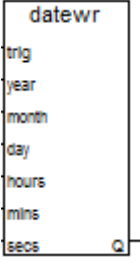
There are a number of functions that are able to read and write the Real Time Clock in several different formats.

Write The Time To The RTC From Integers -- Timewr()

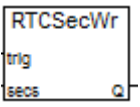
<i>Type</i>	C Function
<i>Function</i>	Write the system time (without the date).
<i>Syntax</i>	ANA TimeWr(ANA trigger, ANA hours, ANA minutes, ANA seconds);
<i>FBD</i>	
<i>Parameters</i>	trig: ANA zero/non-zero triggers the actual write on a transition from zero to non-zero hours: ANA 0-23 the current hour (24 hour format) mins: ANA 0-59 the current minute secs: ANA 0-59 the current second
<i>Return Value</i>	Q: BOO follows the "trigger" input.
<i>Remarks</i>	
<i>ST Example</i>	<pre>(* Set the time when triggered *) timewr(trigger, hours, minutes, seconds); if not (trigger = 0) then trigger := 0; (* reset the trigger *) end_if;</pre>

Write The Date And Time To The RTC From Integers -- Datewr()

Type	C Function
Function	Write the system date and time.
Syntax	BOO DateWr(BOO trigger, ANA year, ANA month, ANA day, ANA hours, ANA minutes, ANA seconds);

FBD	
Parameters	<p>trig: BOO triggers the actual write on a transition from false to true</p> <p>year: ANA 1980-2079 the current year</p> <p>month: ANA 1-12 the current month (1=January)</p> <p>day: ANA 1-31 the current day of the month</p> <p>hours: ANA 0-23 the current hour (24 hour format)</p> <p>mins: ANA 0-59 the current minute</p> <p>secs: ANA 0-59 the current second</p>
Return Value	Q: BOO Follows the "trigger" input
Remarks	The 'trig' input must return to the false state to retrigger another write. Using the 'true' constant will not result in writing the real time clock more than one time at startup.
ST Example	<pre>(* Set the date & time when triggered *) DateWr(trigger, year, month, day, hours, minutes, seconds); if not (trigger = 0) then trigger := 0; (* reset the trigger *) end_if;</pre>

Write The Current RTC Seconds Since 00:00 01/01/70 -- RTCSecWr()

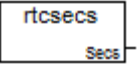
Type	C Function
Function	Write the system date and time in seconds since Midnight on January 1st, 1970.
Syntax	BOO RTCSecWr(BOO trigger, ANA seconds);
FBD	
Parameters	<p>trig: BOO FALSE/TRUE triggers the actual write on a transition from zero to non-zero</p> <p>secs: ANA 0-4294967295 the current second since Midnight 01/01/1970</p>
Return Value	Q: BOO Follows trig input.
Remarks	Trigger input must set to false and true again to trigger a second write.

<i>ST Example</i>	<pre>(* Set the date & time when triggered *) RTCSecWr(trigger, seconds); if trigger then (* reset the trigger *) trigger := FALSE; end_if;</pre>
-------------------	---

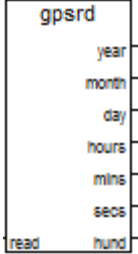
Read The RTC Into Integers -- DateRd()

<i>Type</i>	C Function Block
<i>Function</i>	Read the current system date and time into individual integer registers.
<i>Syntax</i>	DateRd();
<i>FBD</i>	<p>The diagram shows a rectangular function block labeled 'daterd' at the top. On the right side of the block, there are six output terminals, each with a label: 'year', 'month', 'day', 'hours', 'mins', and 'secs' from top to bottom.</p>
<i>Parameters</i>	None
<i>Return Values</i>	<p>year: ANA 1980-2079 the current year</p> <p>month: ANA 1-12 the current month (1=January)</p> <p>day: ANA 1-31 the current day of the month</p> <p>hours: ANA 0-23 the current hour (24 hour format)</p> <p>mins: ANA 0-59 the current minute</p> <p>secs: ANA 0-59 the current second</p>
<i>Remarks</i>	To use the function block in structured text , you must declare a function block instance for each operation used.
<i>ST Example</i>	<pre>(* read the current date and time *) DateRd(); year := DateRd.year; month := DateRd.month; day := DateRd.day; hours := DateRd.hours; minutes := DateRd.mins; seconds := DateRd.secs;</pre>

Read The Current RTC Seconds Since 00:00 01/01/70 -- RTCSecs


<i>Type</i>	C Function C Function Block
<i>Function</i>	Returns the number of seconds since midnight Jan 1 1970.
<i>Syntax</i>	ANA RtcSecs();
<i>FBD</i>	
<i>Parameters</i>	None
<i>Return Value</i>	Secs: ANA Seconds since midnight Jan 1 1970.
<i>Remarks</i>	Provides a means of accessing the real time clock raw seconds value.
<i>ST Example</i>	(* Read Seconds Since Jan 1 1970 *) seconds := RtcSecs(true);

Read The Time And Date From GPS -- gpsrd()

<i>Type</i>	C Function Block
<i>Function</i>	Returns current date and time values from a GPS device.
<i>Syntax</i>	ANA gpsrd(BOO read)l
<i>FBD</i>	
<i>Parameters</i>	read: BOO Set to TRUE to cause date and time outputs to update. Setting to FALSE will make outputs hold last value.
<i>Return Values</i>	year: ANA (1970-2069) The current year. month: ANA (1-12) The current month (1 = January). day: ANA (1-31) The current day of the month. hours: ANA (0-23) The current hour (24 hour format). mins: ANA (0-59) The current minute. secs: ANA (0-59) The current second. hunds: ANA (0-99) The current hundredths of a second.


<i>Remarks</i>	<p>Returns current date and time values from a GPS device that has been connected to and configured for the controller. When READ is TRUE the outputs will update every scan. When READ is FALSE the outputs will hold the last value. The date and time will also be adjusted for the local timezone and daylight savings flag that have been specified for the controller. See <i>Global Positioning Satellite (GPS) Interface</i> (on page 497) for more information.</p> <p>To use the function block, you must declare a function block instance for each operation used.</p>
<i>ST Example</i>	<pre>(* gps is a function block instance of gpsrd() *) (* read the current GPS time *) gps(read); (* if we have read then store off the new clock values *) if read then years := gps.year; months := gps.month; days := gps.day; hours := gps.hour; minutes := gps.minute; seconds := gps.second; hundredths := gps.hunds; (* reset the read flag so we can toggle the read action again *) read := FALSE; end_if;</pre>

Read Date, Time Or Day Of Week -- day_time

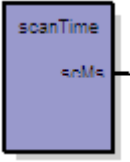
<i>Type</i>	C Function
<i>Function</i>	<p>ISaGRAF Legacy function</p> <p>Gives date or time of the day as a message string.</p>
<i>Syntax</i>	STRING DAY_TIME(DINT sel)
<i>FBD</i>	
<i>Parameters</i>	<p>SEL: DINT output selection</p> <p>0= get current date</p> <p>1= get current time</p> <p>2= get day of week</p>

<i>Return Values</i>	Q: STRING returns different time information strings based on SEL input.
<i>Remarks</i>	This function is not affected by ICL's Time Zone setting or Daylight Savings flag Legacy ISaGRAF function.
<i>ST Example</i>	(* Display text format is: 'YYYY/MM/DD ; HH:MM:SS' *) Display := Day_Time (0) + ' ; ' + Day_Time (1);

Get The Maximum Scan Time -- scanmax()

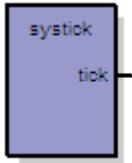
<i>Type</i>	C Function
<i>Function</i>	This function outputs a 32bit value that represents the number of milliseconds for the maximum scan since application start.
<i>Syntax</i>	DINT SCANMAX();
<i>FBD</i>	
<i>Parameters</i>	None
<i>Return Values</i>	mxMs: DINT scantime Maximum number of milliseconds of all scans of the current runtime.
<i>Remarks</i>	None
<i>ST Example</i>	(* Get the maximum scan time since application restart *) scanMaxMs := SCANMAX();

Get The Scan Time Of The Previous Scan -- scantime()

<i>Type</i>	C Function
<i>Function</i>	This function outputs a 32bit value that represents the number of milliseconds for the previous scan.
<i>Syntax</i>	DINT SCANTIME();
<i>FBD</i>	
<i>Parameters</i>	None
<i>Return Values</i>	scMs: DINT scantime Number of milliseconds of previous scan.
<i>Remarks</i>	None

<i>ST Example</i>	(* Get the scan time of the previous scan *) scanMs := SCANTIME();
-------------------	--

Get The Number Of Milliseconds Since Startup -- systick()

<i>Type</i>	C Function
<i>Function</i>	This function outputs a 32bit value that represents the number of milliseconds since the controller has started up.
<i>Syntax</i>	DINT SYSTICK();
<i>FBD</i>	
<i>Parameters</i>	None.
<i>Return Values</i>	tick: DINT Number of milliseconds since power on.
<i>Remarks</i>	None
<i>ST Example</i>	(* Get the number of milliseconds since system startup*) ticksMs := SYSTICK();

Redundancy Function Block For Legacy Controllers

The redundancy feature requires two controllers with the same program to work. The system consists of one master and one slave unit with shared local or networked I/O. The two units are connected via a null modem RS-232 cable.

When a controller is configured as a master, a heart beat timer set point sets the period in which the master attempts to communicate with the slave. The slave responds and both systems post an error code of "0" indicating all is well.

After verifying that the slave is present, the master will send all of its data to the slave at the period specified by the data timer set point.

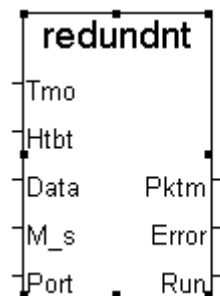
If the master fails to send a heartbeat or data message for the period of the timeout set point, the slave will assume the master is in a failed condition and take over control until such time as the master comes back on-line.

After a failure and changeover to the slave unit, the master must reestablish communication for a minimum of three heartbeats before slave transfers its data to the master and relinquishes control. The master will resume periodically sending data to the slave with heartbeats interspersed between transfers.

Redundnt Function Block

The redundancy feature is accomplished by the use of the Redundnt function block. This block is used to configure and control all redundancy related features of the master-slave system. The function block may be implemented in any ISaGRAF language and is shown in this document as implemented in Function Block Diagram (FBD).

Arguments



Tmo ANA

Time period in milliseconds to wait for a response from slave before posting an error. Used as response delay on master side. On the slave side, this time is used to configure the time before failover. If the slave has not received a heartbeat or data transfer from the master in this amount of time, the slave will go into failover mode. This time can be gauged from the Pktm (peak time) output.

Htbt ANA

Time period in milliseconds to send heartbeat from the master. If heartbeat is not received within the specified Tmo time on the slave side, the master is considered to be in failure and the slave will take over control.

Data ANA

Time period in seconds to send data to slave. Used on master side only. Sending data can take a significant amount of scan time. Use care when setting this parameter. It is recommended to set this time larger than the Htbt time to prevent the system from spending all its time transferring data.

M_s BOOL

Set true if master side controller. Set false if slave side controller. It is recommended that the READ_SW function be used in conjunction with the hardware switches on the controller to determine which controller is master and which is the slave.

Port ANA

Comport number to transmit and receive data. Com3 through Com6 may be used. The comport must be the same on both master and slave controllers. Com1 and Com2 (if available) may not be used for this function.

Return Values

Run BOOL

This output controls execution of code within the Master/Slave program. It can control the use of Return statements (see Program Implementation below), the IO_CTRL function block to turn on and off local I/O and, the NS_CTRL function block to turn on and off Network Sessions for networked I/O.

Err ANA

This output annunciates the state of the system depending on whether it is a master or a slave. 0 = System Okay or Nominal, 1 = Communications lost to other unit, and 2 = Bad Data has been received or refused from the other unit.

The following table shows the states of the error and run outputs based on whether we are a master or a slave.

	Master		Slave	
Err	Err Meaning	Run	Err Meaning	Run
0	System okay	True	System okay	False
1	Communication lost to slave	True	Communication lost to master	True
2	Bad Data	True	Bad Data	False

Pktm ANA

This output shows the longest transaction time since restarting the controller (Peak Transaction Time). It can be used as a diagnostic tool to set the Tmo input parameter. The Tmo should be slightly greater than the Pktm under normal, non-failover conditions.

Technical Considerations

Only ports with 64 or 32 byte FIFO's may be used. These include ports Com3 through Com6. Com1 and Com2 do not have sufficient buffers to maintain a constant flow of data.

Internal data from ISaGRAF function blocks and functions gets transferred with the database.

The larger the system database, the longer updates will take. Choose wisely when configuring data update and heartbeat times.

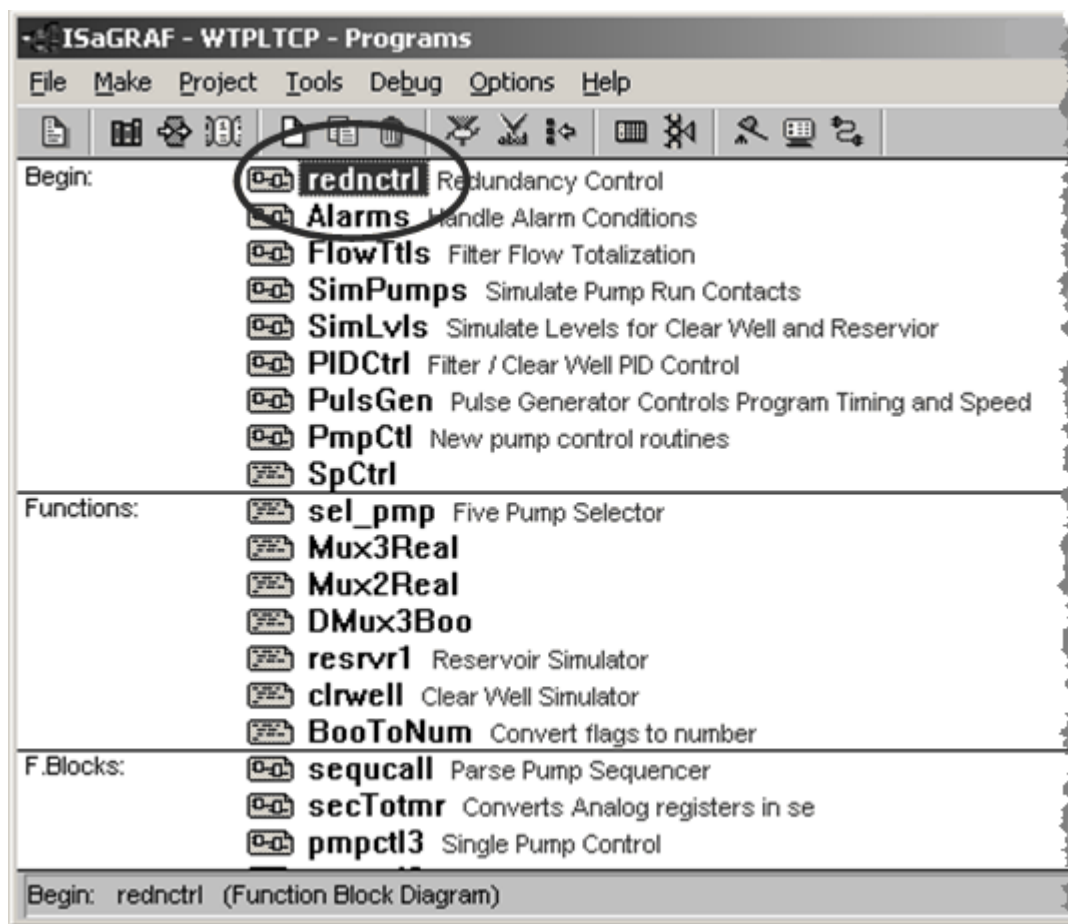
For management reasons, the Redundnt block itself does not handle enabling or disabling the execution of code. The program must be arranged in such a way to utilize the Run output to decide what code will execute and what will not. It is suggested practice to separate the code that gets executed under failover management from the code that will run in the controller full-time.

Memory usage is of prime concern. The database must be duplicated in order to maintain synchronization to the slave controller. This duplication can cause a near doubling of memory usage. A snapshot of all values must represent the system at a given time and interval. All data will be transmitted as a block or a set of blocks but the snapshot will represent a given moment of execution.

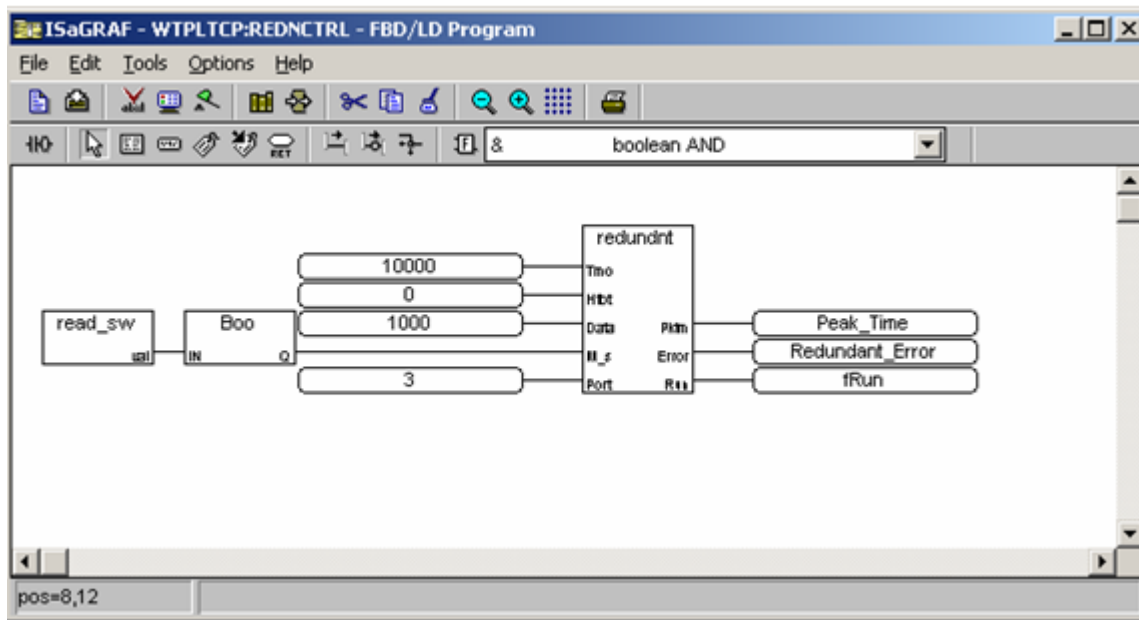
Program Implementation

This MUST be the first block in the PLC cycle within an ISaGRAF program at the start of the Begin section. One controller is configured as a master and one as a slave using the hardware dip-switch.

The following is a screen capture from a working control program with Redundnt block implemented in the first block.

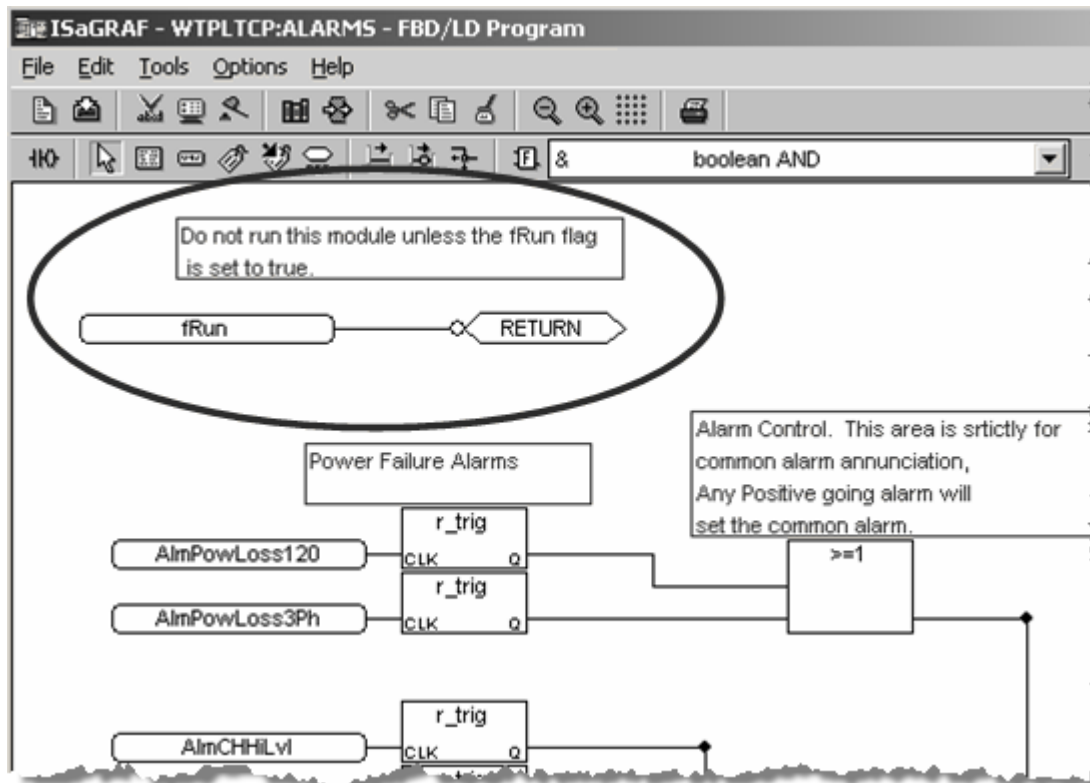


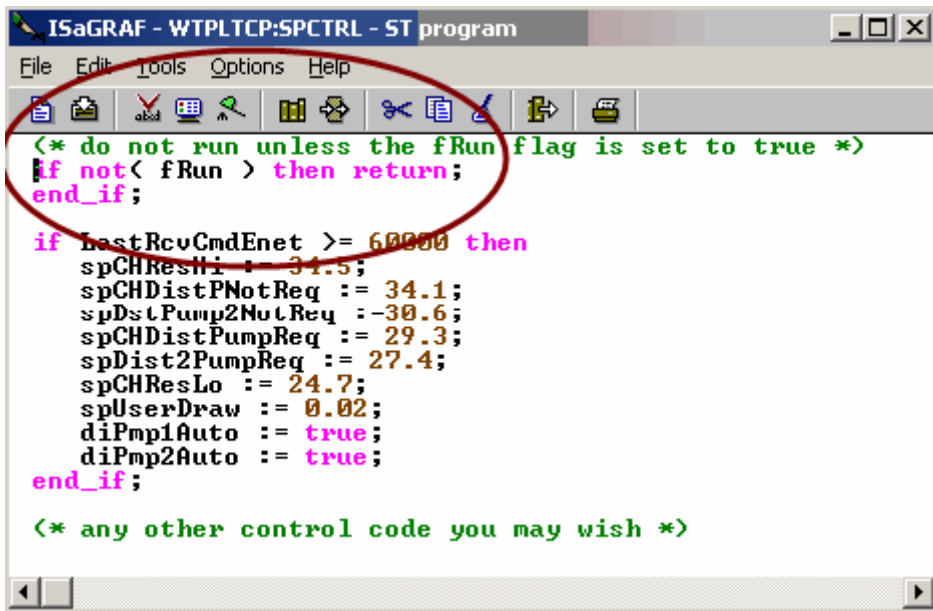
The redundancy block is implemented in the initial Begin section program module “rednctrl”. The rednctrl block is implemented as the following:



The three variables on the right side of the Redundnt block are the only variables not affected by the Master/Slave database transfer. They are always set after the Redundnt block executes. The fRun flag is used to make the decisions of what code is run when the Master or the Slave takes over operation.

To implement the fRun flag in the program, each module must be bypassed when the fRun flag is set to FALSE (do not run). The Return (do not execute) decision is shown in Function Block as well as Structured Text below:





```

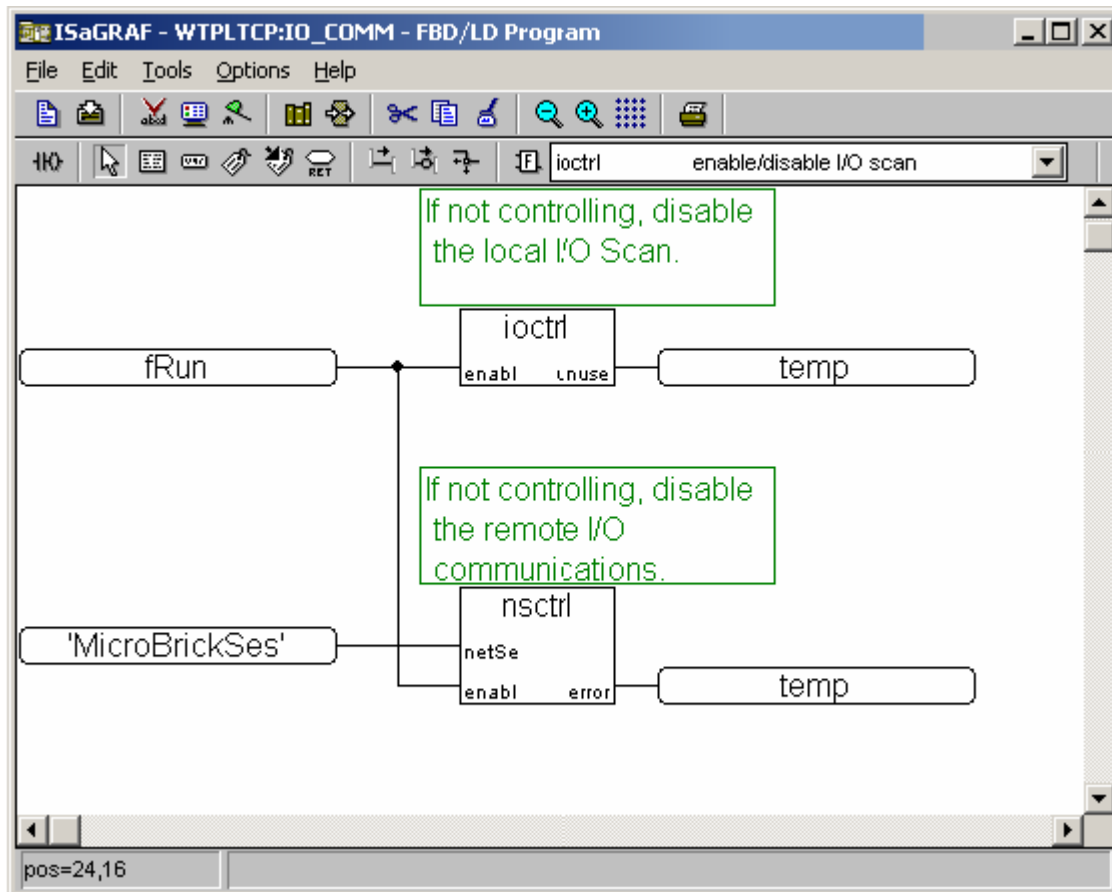
ISaGRAF - WTPLTCP:SPCTRL - ST program
File Edit Tools Options Help
(* do not run unless the fRun flag is set to true *)
if not< fRun > then return;
end_if;

if bestRcvCmdEnet >= 60000 then
  spCHResHi := 34.5;
  spCHDistPNotReq := 34.1;
  spDistPump2NotReq := 30.6;
  spCHDistPumpReq := 29.3;
  spDist2PumpReq := 27.4;
  spCHResLo := 24.7;
  spUserDraw := 0.02;
  diPmp1Auto := true;
  diPmp2Auto := true;
end_if;

(* any other control code you may wish *)

```

The fRun flag may also be used to enable and disable the local I/O scan on the controller and turn on and off Network Sessions that communicate with remote I/O. The following is an example of this implementation:



Deployment

Deploying an application, especially in an Ethernet environment can be a little tricky. Although each controller has the same program, it is likely that both will reside on the network and need different IP addresses in order to function correctly.

You must first have one application configured as described in the Program Implementation section of this document. It is recommended that all deployment take place over the serial download interface rather than Ethernet. If desired, configure the “master” or original application to debug over Ethernet in the **Node | Options ISaGRAF** tab.

Connect the two controllers you wish to setup as a redundant pair with a null modem cable between the desired comports configured by the Port parameter of the Redundnt block.

Connect a null modem cable from the development computer to the “master” controller. Set the master controller’s dipswitch to “1” (switch 1 through 7 off and 8 on). Set the “slave” controller’s dip switches to all switches off.

From ScadaBuilder, select the Target | Send Complete Controller Setup menu option and let the system complete the entire download.

Test your application including communications, I/O, Ethernet communication and control to make sure it functions as intended when the redundant slave is not present.

After verifying all functions, change the IP address in the **Node | Options TCP/IP** tab to allow both the master and the slave to coexist on the same network.

Move the serial download cable to Com1 of the slave controller and select the Target | Send Complete Controller Setup... menu. Allow the download to complete and then restart the slave.

General Notes.

The above procedure will synchronize all versions of firmware on the controllers which **MUST** be done in order for redundancy to work. Version differences in the firmware can cause application mismatch errors that annunciate in the system.log file on the controllers.

Once the above procedure has been completed, both units may be debugged over Ethernet to allow for fail over checking and testing. Only the IP address needs be changed before debugging to a second unit. This will not affect the program version information or the transferred data.

Redundancy Function Block for Pinnacle Controllers

The redundancy feature requires two controllers with the same program to work. The system consists of one Master and one Slave unit with shared local or networked I/O. The two units may be connected via Ethernet, Serial (RS-485 is supported) or both interfaces; one Ethernet and the other Serial with the Ethernet port acting as a primary and the Serial port acting as a redundant backup.

When a controller is configured as a Master, a heart beat timer set point sets the period in which the Master attempts to communicate with the Slave. The Slave responds and both systems post a status code of "0" indicating all is well.

After verifying that the Slave is present, the Master will send all of its data to the slave at the period specified by the data timer set point. In the case of an Ethernet interface, the heartbeat is only used after there is a link failure. Otherwise the Ethernet interface will simply send the data as fast as it can according to the IP Transfer time.

If the Master fails to send a heartbeat or transfer application data for the period of the timeout set points, the Slave will assume the Master is in a failed condition and take over control until such time as the Master comes back on-line.

After a failure and changeover to the Slave unit, the Master must reestablish communication with a heartbeat before Slave transfers its data to the Master and relinquishes control. The Master will resume periodically sending data to the Slave.

When Ethernet and Serial are used at the same time, the Master will simply send a heartbeat over the Serial link to verify that it is still functional. If the Ethernet link is found to be non-function but the Serial link is still working, the Master will transfer over the Serial link although these transfers will be much slower. When the Serial link is being utilized as a backup, the Master will test the Ethernet link to see if it has come alive again using the heartbeat/heartbeat timeout functionality. When it receives a response from the Slave over that link, it will resume using Ethernet for transfers and will resume sending heartbeats over the Serial link.

Setup And Technical Considerations

Hardware

Controller models must be the same (Everest to Everest for example).

Redundancy can be implemented using Ethernet or Serial (RS485 supported) or a combination of the two working with Ethernet as the primary interface and Serial as the backup (redundant) interface.

Over IP (Ethernet), UDP port 52230 is used to transmit heartbeats to gauge IP (Ethernet) link quality and transfer application data snapshots. A direct Ethernet connection may be used (one cable from the Master controller to Slave controller) or connection may be accomplished through an Ethernet switch or hub.

Over Serial, a fixed baudrate of 115200bps is used to transmit heartbeats to gauge link quality and to transfer application snapshots with multiple blocks of data per transfer. A serial null modem cable must be used if connecting two serial ports directly. An RS-485 two wire link (Com 2 on Pinnacle series controllers) may used as well.

When Serial and IP are used together and both links are working, heartbeat data is used on the Serial link with the IP link active and data transfers over Serial will only take place when the IP link is down for any reason. When the IP link is down, heartbeats are sent out the IP interface until a response is received from the Slave and IP operation resumes.

Data Transfers

Internal data from ISaGRAF function blocks gets transferred with the database along with all registers, arrays and structures defined within the ISaGRAF dictionary.

The larger the system database, the longer updates will take. Choose wisely when configuring data update and heartbeat times. Peak Transfer Time and Last Transfer Time statistics are provided to help in the tuning process to maximize data transfer times and application recovery.

For management and database synchronization reasons, the Redundant block itself does not handle enabling or disabling the execution of code. The program must be arranged in such a way to utilize the Run output to decide what code will execute and what will not. It is suggested practice to separate the code that gets executed under fail over management from the code that will run in the controller full-time.

A Boolean output is provided to tell the logic whether the unit is running or not.

A second Boolean output is provided to tell the logic whether this unit is the Master or the Slave.

A Status integer is provided to give a bit map of all current errors.

The database must be duplicated in order to maintain synchronization to the Slave controller sending snapshots of the database in a static state at the time of transfer. A snapshot represents all values at a given time and interval. All data will be transmitted as a block or a set of blocks but the snapshot will represent a given moment of execution (a.k.a. a single scan of the PLC cycle).

Redundant Block Inputs

The inputs of the Redundant block must be constants or defined words.

DO NOT USE REGISTERS FOR ANY INPUTS OF THE REDUNDANT BLOCK!

Registers will be overwritten at transfer time. The only registers that are not overwritten are the outputs of the Redundant block itself. All other data is overwritten from the transfer.

Local I/O

Using local I/O can be tricky. Because Analog Inputs are often current (4-20mA) inputs, they cannot be connected in parallel. Both units should probably utilize a voltage mode for reading sensors with an external pass resistor if current is used so the current input may be used in parallel. Resistive type measurements such as potentiometers, thermistors, and RTDs cannot be used in parallel and convertor isolators should be used--these would preferably output a voltage that the both controllers can read.

In most cases, Digital Inputs can be run in parallel.

In most cases, Digital Outputs can be run in parallel so long as they use the same power source. If they need to use redundant supplies then slave relays should be employed to do any switching.

In no case can Analog Outputs be used in parallel. They will need to be connected through isolators or slave relays that will switch them to the appropriate inputs on remote units.

The local I/O can be disabled when the Master or Slave is inactive (Run output is FALSE).

Remote I/O

Remote I/O may be utilized but should be accessed either via Ethernet (Modbus TCP/IP for example) or from an multi-drop network such as RS-485. ICL manufactures a serial splitter for RS-232 interfaces where two controllers can access a single RS-232 port on a remote I/O unit. The one requirement is that the protocol used be half-duplex only.

Network Sessions using the port should be disabled when the Master or Slave is not in Run mode (Run output from the Redundant block is FALSE).

PLC Scans And Application Updates

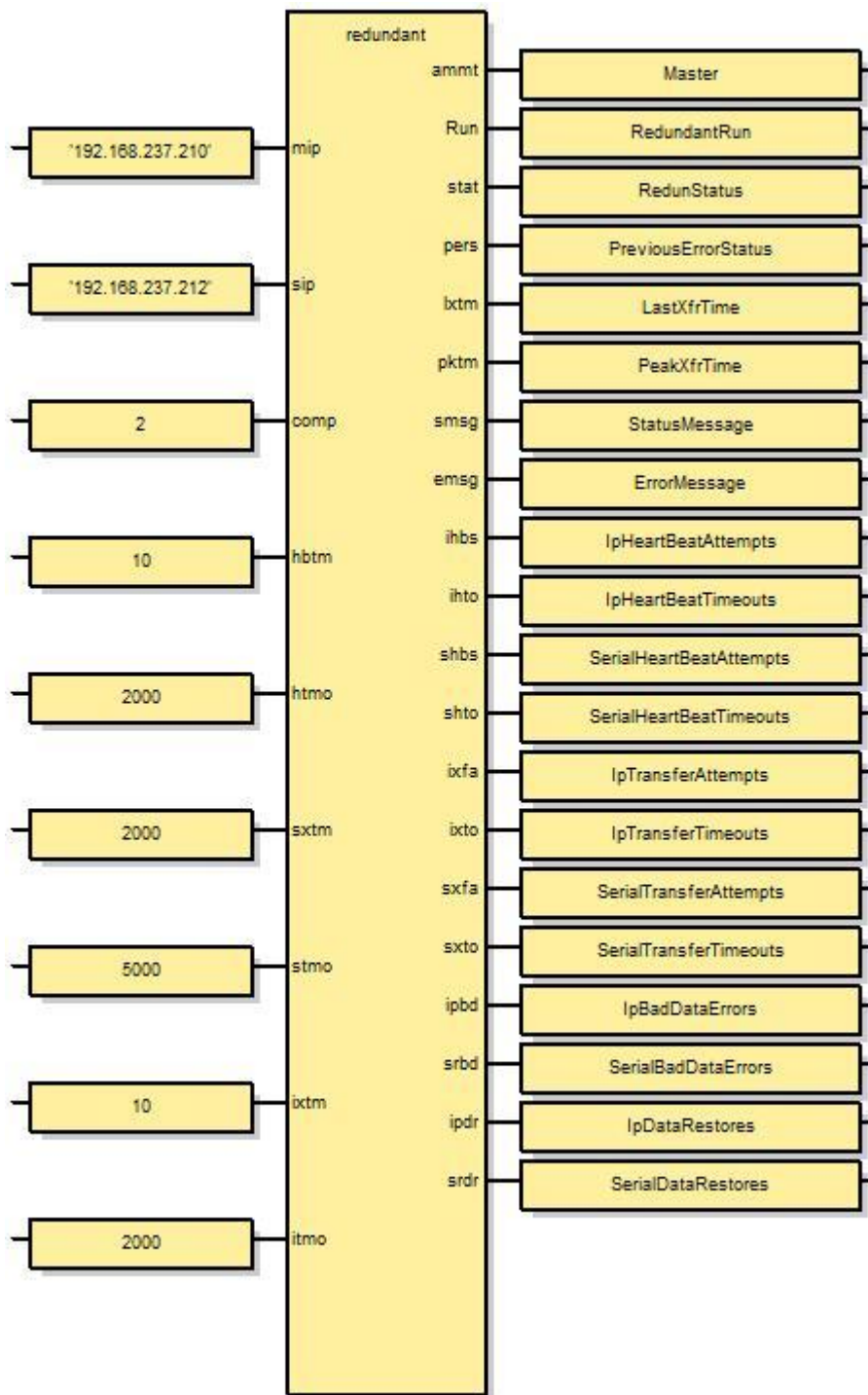
When using IP (Ethernet), the update from the Master to Slave can be tuned to a very fast pace. It takes two PLC scans to send a snapshot of the database to the remote unit.

When using IP (Ethernet) with Serial backup, it takes a maximum of 4 PLC scans to transfer the data over IP. The system must intersperse Serial heartbeats (2 PLC cycles) while transferring the data over IP (2 more PLC cycles) for a possible maximum of 4 PLC cycles if all timing is tuned as tight as possible. If the Serial link fails the heartbeat test, the transfers over IP will slow down by the Heartbeat Timeout time. Slave fail over (Run flag set to TRUE on the Slave) only occurs when both interfaces have timed out.

When using Serial only as the Redundant link (or using Serial fail over mode when IP and Serial are used together and IP is in failure), transfers take a much greater number of scans and time as the Serial link must transfer multiple blocks of data (one block every 2 PLC cycles) to complete the transfer. A block size of 1462 bytes is used for each block until the transfer is finished. The number of blocks varies by application size. The Slave does not care which interface it receives data on. Either one or both active verifies that the Master is up and running.

Redundant Function Block Parameters

The redundancy feature is accomplished by the use of the Redundant function block. This block is used to configure and control all redundancy related features of the Master-Slave system. The function block may be implemented in any ISaGRAF language and is shown in this document as implemented in Function Block Diagram (FBD). This is a typical high speed configuration with the Serial link (Com 2 RS-485) as a backup link. 192.168.237.210 is the Master here and 192.168.237.212 is the Slave.



Input Parameters

DO NOT USE REGISTERS FOR ANY INPUTS OF THE REDUNDANT BLOCK!

The inputs of the Redundant block must be constants or defined words.

They will be overwritten at transfer time. The only registers that are not overwritten are the outputs of the Redundant block itself. All other data is written from the transfer.

mip - MasterIP STRING(20) - Master IP Address is used to determine that this unit is the Master by matching the IP address to the IP address configured in the **Node Settings | Target Configuration tab | Ethernet** tab of the controller. If it is a match, then the ammt output is set to true and this unit acts as the Master. If a unit's IP address does not match, then the non-matching unit is the Slave.

sip - SlaveIP STRING(20) - Slave IP Address if configured enables the use of the IP link as the primary transfer medium. If this string is empty, then the IP link is disabled and the serial link must be configured (non zero in the comp parameter). If neither the Slave IP or the Comport are configured, an error is posted and the Redundant block is disabled.

comp - Comport DINT - Serial Interface Port Number configures the Serial communications port to be used as either the primary link (no Slave IP is configured) or as the secondary interface (Slave IP is configured). A value of zero disables the Serial Link altogether. If neither the Slave IP or the Comport are configured, an error is posted and the Redundant block is disabled.

hbtm - HeartBeatTime DINT - Heart Beat Time mS is the number of milliseconds to wait before initiating a heartbeat command (either for the Serial link or for a failed IP link). Heartbeats are not used for the IP link unless it has failed.

htmo - HeartBeatFailTimeout DINT - Heart Beat Fail Timeout mS

Master - time to wait after a heartbeat command is sent to post a failure for that interface.

Slave - the Slave uses HTMO parameter times two (2x) as the time to wait when no communications has been received to post a comm fail and set its Run output to true. If both IP and Serial links are used, communications over either link will reset this timeout and continue normal operation.

sxtm - SerDataXfrTime DINT - Serial Data Transfer Interval Time mS sets the interval at which application data is transferred over the Serial Link in the Master only. When both IP and Serial links are enabled, this timer is only used when the IP link has failed.

stmo - SerDataXferTimeout DINT - Serial Data Transfer Fail Timeout mS is used to gauge a timeout during a Serial link application transfer. It is used to gauge a failure on both the Master and the Slave after a transfer has been initiated. If a Serial transfer is interrupted, either the Master or the Slave will reset operations when the timeout expires.

ixtm - IPDataXfrTime DINT - IP Data Transfer Interval Time mS is the interval in milliseconds that the Master will initiate a transfer over the IP link. This parameter is not used on the Slave.

itmo - IPDataXferTimeout DINT - IP Interface Data Transfer Fail Timeout mS is used to gauge an interruption of a transfer over IP in both the Master and the Slave. If an IP transfer is interrupted, either the Master or the Slave will reset operations when the timeout expires.

Runtime Output Parameters:

All output parameters of the Redundant block are NOT overwritten after a successful transfer. They are internal values of the Redundant function block and cannot be retained. They will be reset when the controller application is restarted or power to the controller is cycled.

ammt - AmMaster BOOL - I Am Master (true = Master, false = Slave) - This bit may be used in logic to control behavior on the Slave or the Master accordingly. It can be used to inhibit Alarms, Triggers and other ScadaBuilder logic should such a behavior difference be desired. It is not overwritten on an application transfer.

Run - RunActive BOOL - Run Program(s) On This Unit. This flag must be used to return from functions that should not run when a Slave is not active. It can also be used to disable the local controller's I/O scan and disable any Network Sessions that may communicate to Remote I/O or units in the field.

stat - Status DINT - Output Status Number gives a bit map of the current status of the system including any current errors or configuration errors. If all is well with the system configuration and runtime status, this will return a value of zero. Any non zero value means that an error has happened. The individual bits will tell what has gone wrong.

Multiple bits may be TRUE if more than one error needs to be posted at a time. Presented here are the bits of the error message by bit number and hex (and mask) value :

Runtime Errors:	Master		Slave	
Err And 0x Hex Value	Err Meaning	Run Output	Err Meaning	Run Output
0	System okay	TRUE	System okay	FALSE
Bit 0 or 0x01	Communication lost to slave over Serial	TRUE	Communication lost to master over Serial.	TRUE if Serial only or IP link is down also. FALSE if IP link is still active.
Bit 1 or 0x02	Communications lost to Slave over IP.	TRUE	Communications lost to Master over IP.	TRUE if IP only or Serial link is down also. FALSE if Serial Link is still active.
Bit 0 and 1 or 0x03	Communication lost to Slave over IP and Serial.	TRUE	Communications lost to Master over IP and Serial.	TRUE
Bit 2 or 0x04	Bad Data has been received from Slave.	TRUE	Bad Data has been received from Master.	FALSE so long as the current operation's timeout has not been exceeded.
Bit 3 or 0x08	An attempt to restore application data has come from the Slave but was rejected by the Master because the Master is already in Run mode. This is a result of both Master and Slave running okay but the communications link was totally severed then restored.	TRUE	An attempt to restore application data has come from the Slave but was rejected by the Master because the Master is already in Run mode. This is a result of both Master and Slave running okay but the communications link was totally severed then restored.	FALSE
Bit 4 or 0x10	Master has been restored to running condition after a failure. The Slave has successfully restored data to the Master and gone back to standby.	TRUE	Master has been restored to running condition after a failure. The Slave has successfully restored data to the Master and gone back to standby.	FALSE

Configuration Errors For Both Slave and Master:

All Configuration Errors will turn on bit 13 or 0x2000 hex. If any configuration error is detected the Redundant block is disabled. Configuration errors typically occur the first time the Redundant block is executed at startup.

Err And 0x Hex Value	Err Meaning
Bit 5 or 0x20	There has been a memory configuration error and the application is out of memory.
Bit 6 or 0x40	An IP or Comm port could not be opened (it is already open or bound somewhere in the ScadaBuilder configuration). Comm Port is invalid (must be between 1 and 5) Both IP (Slave IP is blank ") and Comm Port is disabled (set to 0).
Bit 7 or 0x80	Application transfer was attempted but the same application is not on the remote device. Applications must be identical.

pers - PrevErrorStatus DINT - Previous Error Status is stored when an error or change in the error status is detected. It will not be reset when normal operation is restored (stat = 0). The previous error values are defined the same as the STAT output above.

lxm - LastXfrTime DINT Last Transfer Time mS tells how long in milliseconds the last completed transfer of application data took to finish successfully over IP or Serial.

pkm - PeakTime DINT Peak Data Transfer Time mS tell the maximum transfer time for both the IP and Serial links. This maximum is only reset if the application is restarted or the controller has been power cycled. It is useful for setting the IP and Serial transfer timeouts. If it equals the timeouts all the time, a link has either failed or its timeout for that interface is set to low.

smsg - StatusMessage STRING(10) Current Status String Output origin M/S port I/S status is useful for displaying configuration errors and link status (on a TUI or on the User Portal for example). The output is never more than 10 characters so it can fit on the built in LCD HMI's of the Pinnacle controllers. The format is:

<M/S>:<Status>:<IP Link Status>,<Serial Link Status>

<M/S>: I am Master or I am Slave -- M or S

<Status>: Current status -- OK, all is well; RERR, Runtime Error; CERR, Configuration Error

<IP Link Status>: I = IP Link Active Good; i = IP Link Down (RERR will be display in the <Status>)

<Serial Link Status>: S = Serial Link Good; s = Serial Link Down (RERR will be display in the <Status>)

Disposition	Master	Slave
IP and Serial Link Configured and Good.	M:OK :I,S	S:OK :I,S
IP Link Configured and Good	M:OK :I	S:OK :I
Serial Link Configured and Good	M:OK :S	S:OK :S
IP and Serial Link Configured and IP Link Bad but Serial is good	M:RERR :i,S	S:OK :I,S **
IP Link Configured and Bad	M:RERR :i	S:RERR :i
Serial Link Configured and Bad	M:RERR :s	S:RERR :s
IP and Serial Link Configured and Serial Link Bad but IP is good	M:RERR :I,s	S:OK :I,S **
Configuration Error	M:CERR	S:CERR

** Note that the Slave will not post an error until all configured interfaces have failed. The Slave does not discriminate anything but a total failure timeout.

emsg - ErrorMessage STRING(10) Last Error String Output (Origin M/S Port I/S Error). This error string is useful for display the last, most critical error on a TUI (like the local LCD display on the Pinnacles) or other string display device (such as the User Portal web interface). The highest priority message is displayed and lesser errors are bypassed in this string. The string does not reset when the error clears so it give the last highest priority message that has happened. All errors are logged in the unit in the system.log file located on the primary drive, date and time stamped with REDUNDANCY in the log entry.

The format for this message is <origin>:(error)

Possible outputs for this error message are:

Run-Time Errors:

M:CommLoss - Slave side has lost communications to the Master

S:CommLoss - Master side has lost communications to the Slave

M:Restored - Master has recovered from a failure and been restored from the Slave.

M:NoRestore - Link recovered but Master was already running. Application restore from Slave Aborted.

S:Bad Data - Master has received bad data from Slave (transfer aborted or invalid command).

M:Bad Data - Slave has received bad data from Master (transfer aborted or invalid command).

Configuration Errors:

M/S:Mem Error - Master or Slave could not allocate memory for one or more interfaces.

M/S:Port Error - Could not open one of the configured ports or neither IP nor Serial is enabled.

M/S:App Error - Application transfer was attempted but applications in Master and Slave do not match.

ihbs - IpHeartBeatAttempts DINT Number Of IP Heartbeat Attempts (Master Only) - will increment when heartbeats are transmitted over the IP link. This will always increment once at startup to establish communications and will only increment if the IP link has failed and the system is testing for IP communications. Over IP the IP data transfer is used as the heartbeat to increase application snapshots per scan.

ihito - IpHeartBeatTimeouts DINT Number Of IP HeartBeat Timeouts (Master Only) - if an IP heartbeat was attempted and failed (timed out) then this number will increment. If the IP link is disabled, this number will remain at zero.

shbs - SerHeartBeatAttempts DINT Number Of Serial HeartBeat Attempts (Master Only) - if a heartbeat was attempted over the Serial interface, this number will increment. If the Serial interface is disabled, this number will remain at zero.

shto - SerHeartBeatTimeouts DINT Number Of Serial HeartBeat Timeouts (Master Only) - if a Serial heartbeat was attempted and failed (timed out), this number will increment.

ixfa - IpTransferAttempts DINT Number Of IP Transfer Attempts - If the Master attempts an application data transfer over IP, this number will increment on the Master. If a Slave receives an application update command, then this number will increment on the Slave.

ixto - IpTransferTimeouts DINT Number Of IP Transfer Timeouts - If the Master has attempted a transfer over IP and it was interrupted during the process, this number will increment on the Master. If the Slave has received an application update command over IP and the transfer is interrupted, this number will increment on the Slave.

sxfa - SerTransferAttempts DINT Number Of Serial Transfer Attempts - If the Master attempts an application data transfer over Serial, this number will increment on the Master. If a Slave receives an application update command over Serial, then this number will increment on the Slave.

sxto - SerTransferTimeouts DINT Number Of Serial Transfer Timeouts - If the Master has attempted a transfer over Serial and it was interrupted during the process, this number will increment on the Master. If the Slave has received an application update command over Seruak and the transfer is interrupted, this number will increment on the Slave.

ipbd - IpBadDataErrors DINT Number Of IP Bad Data Errors - If the Master or Slave over IP has received data out of sync, or a data transfer has a corrupted CRC, this number will increment.

srbd - SerBadDataErrors DINT Number Of Serial Bad Data Errors - If the Master or Slave over Serial has received data out of sync, or a data transfer has a corrupted CRC, this number will increment.

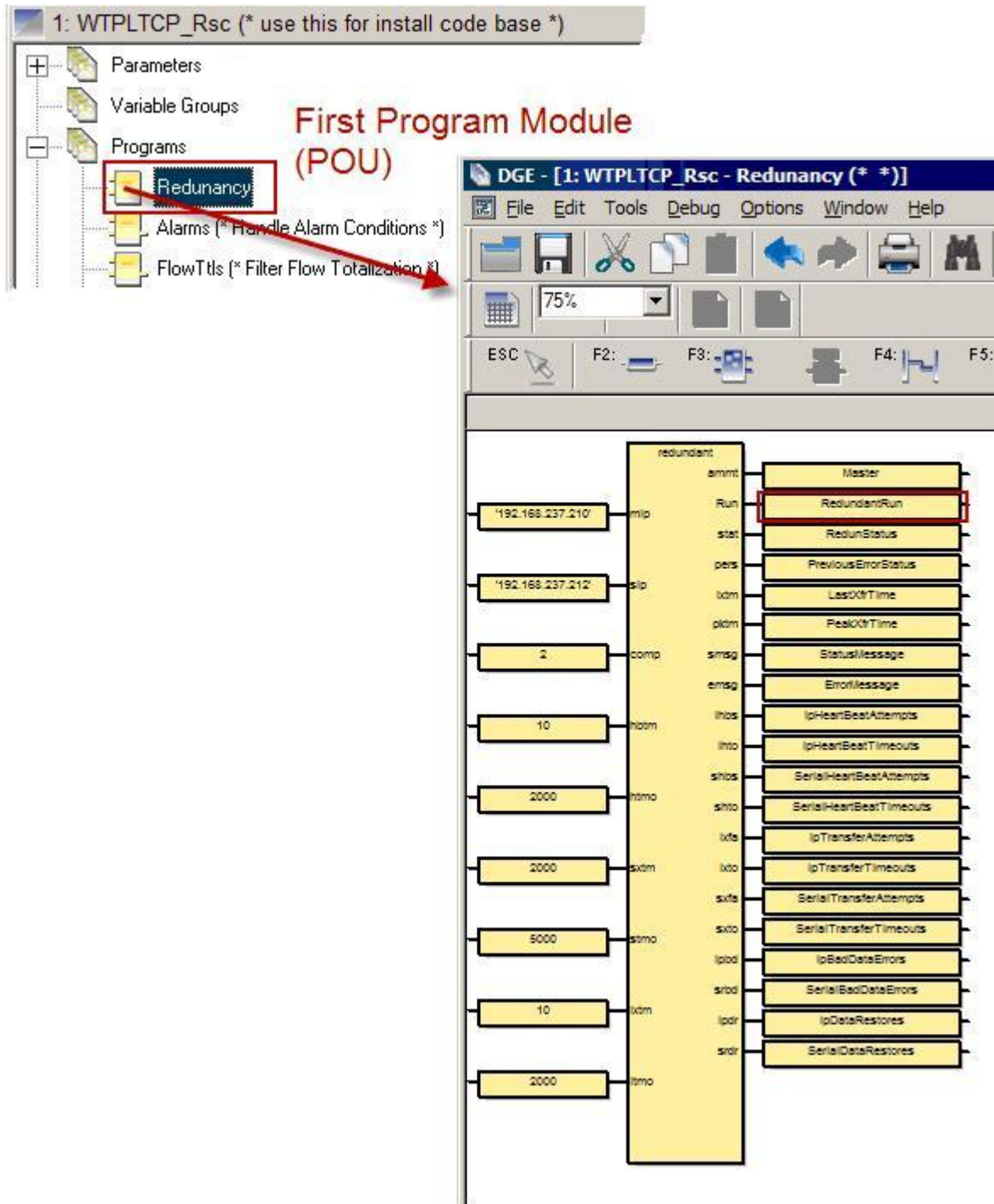
ipdr - IpDataReStores DINT Number of IP Data Store/Restores - If the Master (upon recovery) or Slave (normal operation) has received and verified an application snapshot over IP, then this number will increment.

srdr - SerDataReStores DINT Number Of Serial Data Stores/Restores - If the Master (upon recovery) or Slave (normal operation) has received and verified an application snapshot over Serial then this number will increment.

Program Implementation

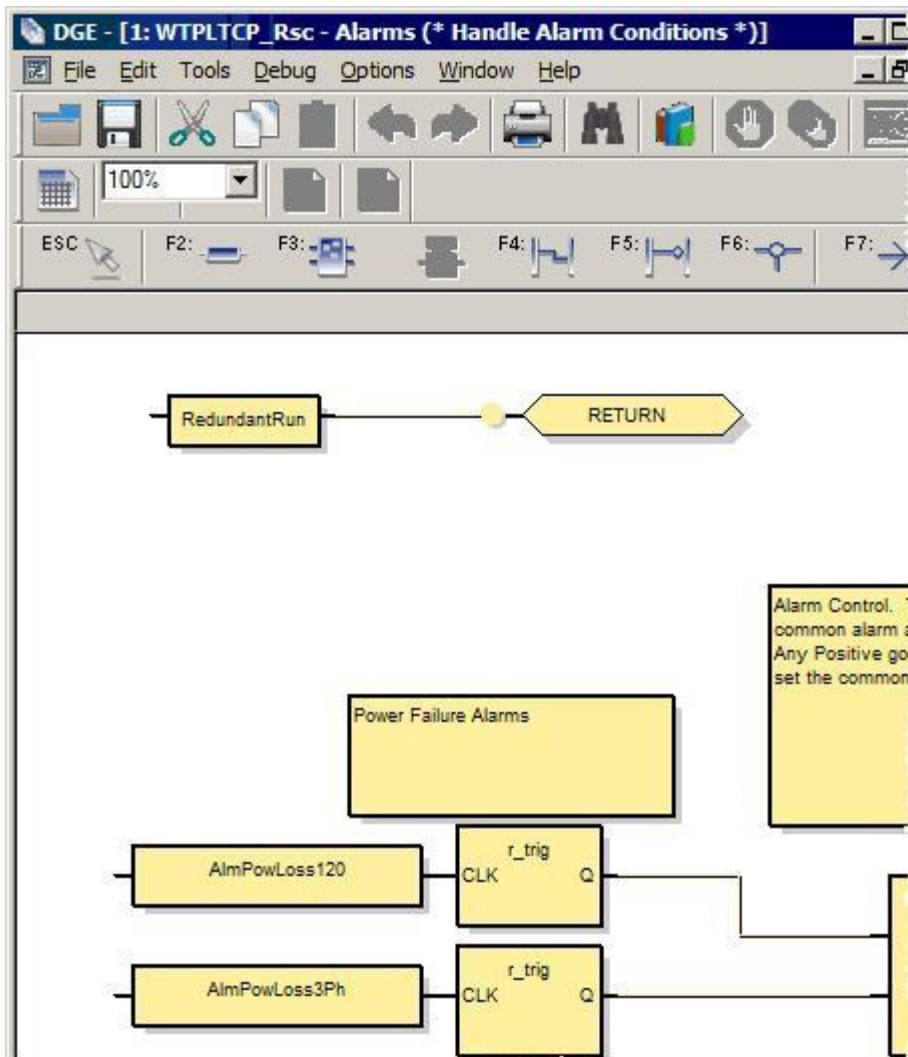
The Redundant function block MUST be the first block in the PLC cycle within an ISaGRAF program at the start of the Programs section. One controller is configured as a Master and one as a Slave using the IP Address of the MIP input to determine which unit is the Master.

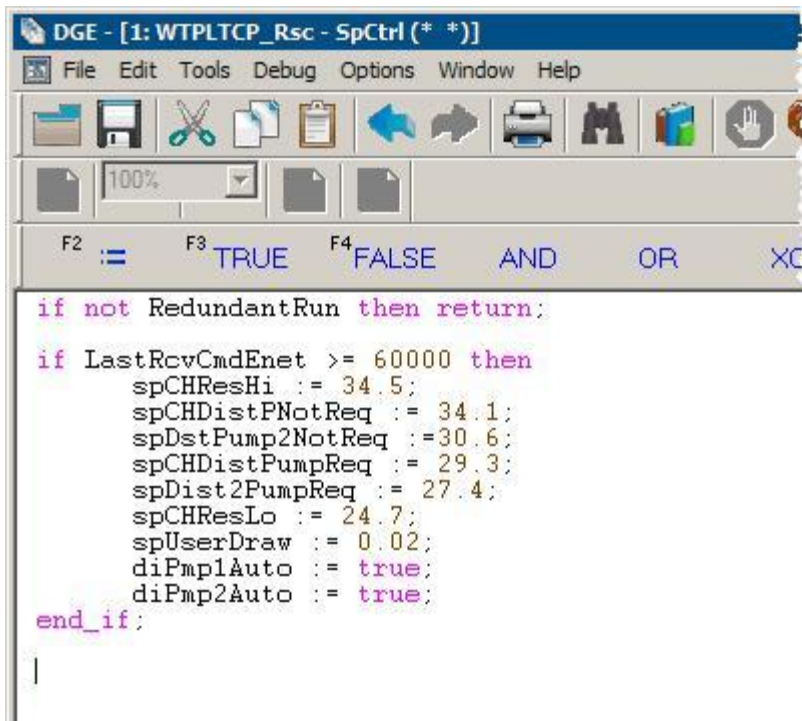
The following is a screen capture from a working control program with Redundant block implemented in the first block.



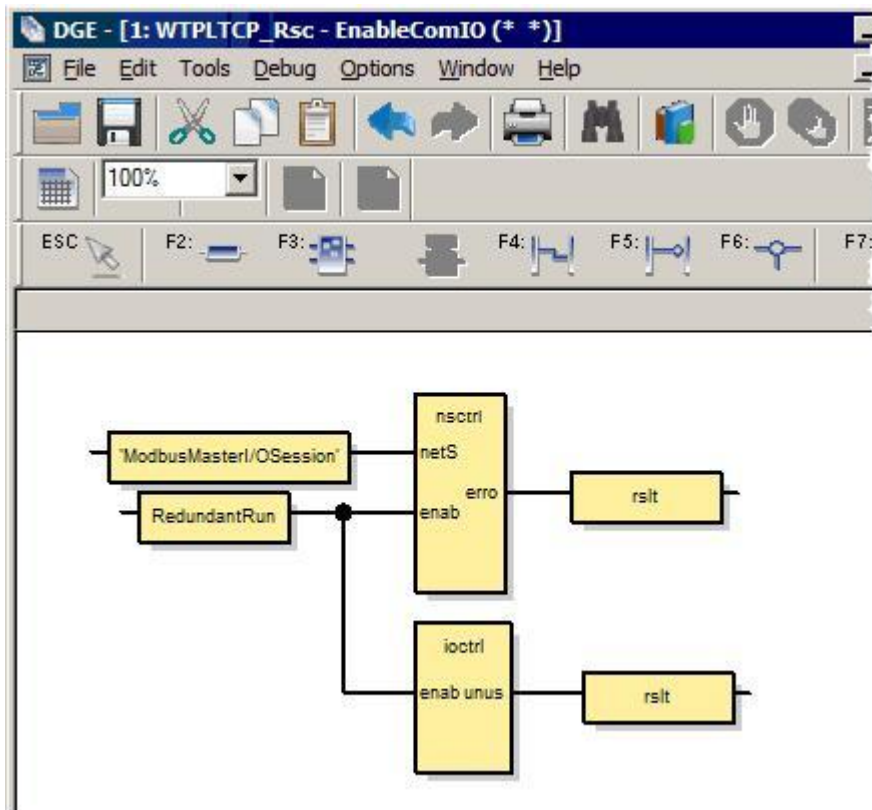
The outputs on the right side of the Redundant block are the only variables not affected by the Master/Slave database transfer. They are always set after the Redundant block executes. The Run output boolean (RedundantRun register in this case) is the used to make the decisions of what code is executed when the Master or the Slave takes over operation.

To implement the Run output boolean in the program, each module in the Programs section must be bypassed when the (RedundantRun flag is set to FALSE (do not run). The Return (do not execute this module) decision is shown in Function Block as well as Structured Text below:





The RedundantRun flag may also be used to enable and disable the local I/O scan on the controller and turn on and off Network Sessions that communicate with remote I/O and remote units. The following is an example of this implementation:



Note: do not bypass this function with RedundantRun return statement or the implementation will not work.

Deployment

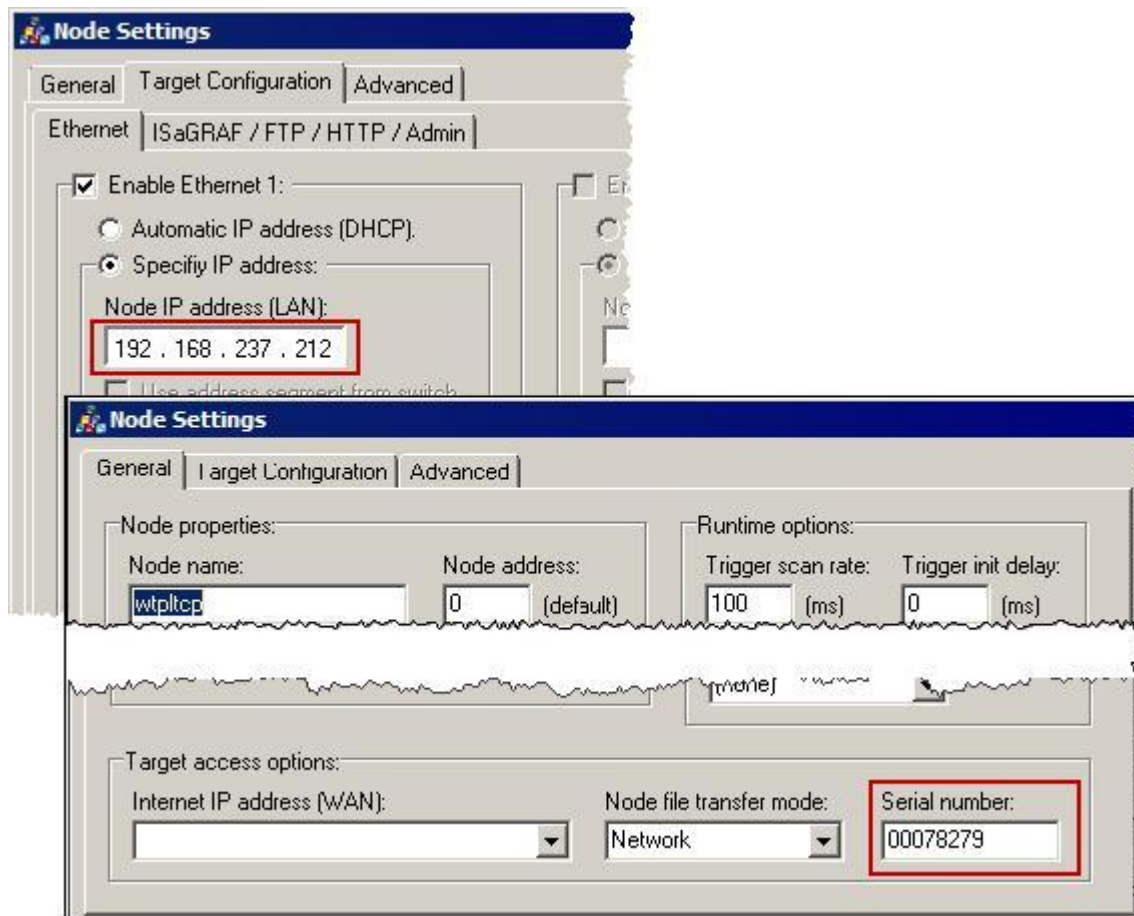
Deploying an application in an Ethernet environment can be a little tricky. Although each controller has the same program, both will reside on the network and need different IP addresses in order to function correctly.

You must first have one application configured as described in the Program Implementation section of this document. Finish the functional testing as if the unit is a Master so that all changes (as much as possible) are finished.

Connect the two controllers you wish to setup as a redundant pair. See *Setup And Technical Considerations* (on page 614) for more details.

From ScadaBuilder, select the Target | Send Complete Controller Setup menu option and let the system complete the entire download to the Master.


Then change the IP address in the Node | Settings | Ethernet tab to the IP address of the Slave and enter the serial number of the Slave in the Node | Settings | General tab:



From ScadaBuilder, select the Target | Send Complete Controller Setup menu option and let the system complete the entire download to the Slave.

Test your application including communications, I/O, Ethernet communication and control to make sure it functions as intended when the redundant Slave is not present.

Connect the Slave and verify that its Run Output goes FALSE once communications is established.

If application updates are needed, simply change the IP address and serial number back to the Master's configuration then download via the Lightning Bolt  in ScadaBuilder. Then change the IP address and serial number back to the Slave's configuration and repeat the Lightning Bolt download.

To debug the units, the same data entry sequence can be repeated for the Slave and Master. A download is not necessary although ISaGRAF and/or ScadaBuilder will tell you that changes have been made (you have after all changed the IP address) and may ask you to rebuild. You can ignore this prompt and just click the Debug button if your IP Address is setup for the unit you want. ScadaBuilder may prompt you to build. Go ahead and do this so that ISaGRAF will take the new IP address. especially if you are switching back and forth between debugging units.

Remember, any changes to the Master **MUST** also be downloaded to the Slave or unpredictable results could occur. If you get into a state where the units are out of sync and won't run, put both units in Loader mode and repeat the download process at the start of this section. That will get the units' downloaded applications back in sync.

ISaGRAF Socket Functions (UDP and TCP/IP)

This section contains basic descriptions for implementing servers and clients with ISaGRAF functions and function blocks.

These functions are available in Pinnacle and later controllers only.

Client Connection Basic Strategy

The basic procedure for creating a client connection (connecting to a listening server) for both TCP/IP and (UDP both Server and Client) are:

- Allocate the socket -- SockAlloc()
- Connected to the TCP/IP address and port of a server -- SockConnect()
- Check the status of the link -- SockConnect()
- Writedata to the server as needed -- SockWrite()
- Check the status of the link -- SockConnect()
- Readdata from the server as needed -- SockRead()
- When done with the connection, terminate the connection one of three ways, -- SockConnect(), SockRelease() or SockFree().

Reconnect as needed and repeat.

An example of this process with extensive comments is available at C:\Program Files\ScadaWorks\Samples\Isa5Sockets\IsaSocketsTCPandUDPClientSample.zbp

From ScadaBuilder do a **Project | Restore** to see the sample code in ISaGRAF 5

Server Implementation Basic strategy

The basic steps for creating a server (listening socket) for TCP/IP are:

- Allocate the socket -- SockAlloc()
- Listen for a connection -- SockListen()
- Check the status of the link -- SockListen()
- Readdata from the server as needed -- SockRead()
- Check the status of the link -- SockListen()
- Writedata to the server as needed -- SockWrite()
- When done with the connection, terminate the connection one of to ways, -- SockRelease() or SockFree().

An example of this process with extensive comments is available at C:\Program Files\ScadaWorks\Samples\Isa5Sockets\IsaSocketsTCPIPServerSample.zbp

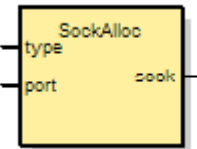
From ScadaBuilder do a **Project | Restore** to see the sample code in ISaGRAF 5



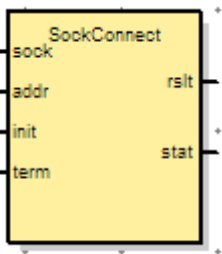
UDP protocol has no connection. There is no reason to implement a listen on a UDP type socket.

Allocate a new socket -- SockAlloc()

Type	C Function
------	------------


<i>Function</i>	Allocates and initialize a socket handle for use with all other socket functions
<i>FBD</i>	
<i>Syntax</i>	DINT SockAlloc(DINT Type, DINT port); returns: DINT SOCK;
<i>Parameters</i>	type: DINT 0 = TCP, 1 = UDP. port: DINT TCP/IP or UDP port number to connect on.
<i>Return Values</i>	sock: DINT Socket handle used by all subsequent socket functions non-zero = good handle 0 = error allocating socket
<i>Remarks</i>	If the socket will not be used after a disconnect, SockFree should be called to free the memory used by the socket. This is done automatically should an ISaGRAF restart occur.
<i>ST Example</i>	<pre> (* allocate the socket to use, set the socket type and port number *) (* SocketType 0 = TCPIP, 1 = UDP *) Socket1Type := 0; SocketHandle := SockAlloc(Socket1Type, 500); (* did we get a valid socket handle? *) if Socket1Handle > 0 then (* go to next state and do a SockConnect *) end_if; </pre>

Connect To a Server or Initialize Socket -- SockConnect()

<i>Type</i>	C Function Block
<i>Function</i>	Establish a connection with a remote host. This function will process the socket as a client and handle the states of a connection already in progress.
<i>FBD</i>	

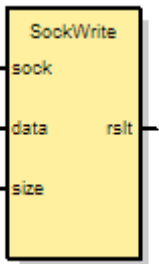
<i>Syntax</i>	SockConnect(DINT SOCK, DINT ADDR, BOOL INIT, BOOL TERM) SockConnect.rslt SockConnect.stat
<i>Parameters</i>	sock: DINT socket to establish connection on. addr: DINT IP address or host name of remote server. init: BOOL set high to initiate a connection when idle. term: BOOL set high to release connection when not idle.
<i>Return Values</i>	rslt: DINT -2 invalid SOCK. -1 = idle (not connected to host). 0 = connected (just connected to host). 1 = active (still connected to host). 2 = release (releasing connection). stat: STRING status of connection.
<i>Remarks</i>	After a connection is initiated, the INIT input must transition low for at least one call to SockConnect() before a connection can be re-established on the socket. SockConnectFBINST is a function block instance of SockConnect declared in the dictionary of ISaGRAF 5. SockConnect should be called and the status checked before every read and write of the socket.
<i>ST Example</i>	<pre> SockConnectFBInst(Socket1Handle, '192.168.3.100', TRUE, FALSE); (* did we get a connection *) if SockConnectFBInst.rslt = 0 then (* go to next connections state and do a read or write of data *) (* see SockRead and SockWrite examples *). end_if; </pre>

Open a Listening Socket To Act As Server -- SockListen()

<i>Type</i>	C Function
<i>Function</i>	Listen for remote connections
<i>FBD</i>	
<i>Syntax</i>	DINT SockListen(DINT Sock); SockListen.Rslt;

<i>Parameters</i>	sock: DINT Socket used for listening for a new connection and monitoring an established connection.
<i>Return Values</i>	rsIt: DINT -2 invalid SOCK. -1 = idle (no client connected). 0 = connected (client just connected). 1 = active (client still connected).
<i>Remarks</i>	<i>SockListen should be called and the status checked before every read and write of the socket. SockListen will handle link status changes every time it is called.</i>
<i>ST Example</i>	<pre> SockListenResult := SockListen(Socket1Handle); (* did we get a connection *) if SockListenResult >= 0 then (* go on and read or write from the connection *) end_if; </pre>

Write Data To a Socket -- SockWrite()

<i>Type</i>	C Function
<i>Function</i>	Write data to an active connection.
<i>FBD</i>	
<i>Syntax</i>	DINT SockWrite(DINT SOCK, STRING DATA, DINT SIZE)
<i>Parameters</i>	sock: DINT connection to write data to. data: STRING data bytes to write size: DINT number of data bytes to write.
<i>Return Values</i>	rsIt: DINT -2 = invalid SIZE (must be <= 255) or invalid SOCK. -1 = no connection established. 0 = success (data written). 1 = blocked (retry call later until complete).

<i>Remarks</i>	<i>SockListen or SockConnect should be called and the status checked before every write of the socket. If the connection is lost then no writes should be done. Under UDP the connection is only monitored for coherence and not whether or not the remote server got messages or maintained a connection. Under UDP, SockWrite will always return success.</i>
----------------	---

ST Example

```

(* check the connection before attempting access *)

(* this check should always be done before attempting to use the connection
*)

(* the remote side could close the socket at any time *)
SockConnectFBInst( Socket1Handle, '192.168.3.100', FALSE, FALSE );
SockConnectResult := SockConnectFBInst.RSLT;

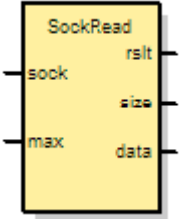
if SockConnectResult = 2 then
    return;
end_if; (* socket may be in process of releasing *)

if SockConnectResult = -1 or
    SockConnectResult = -2 then (* bad return value do not use *)
    return; (* don't attempt to write *)
end_if;

(* write using the same handle as above pass in the buffer and buffer length *)
SockWriteResult := SockWrite( Socket1Handle, TransmitBuffer, MLEN(
    TransmitBuffer ) );
case SockWriteResult of
    (* Success *)
    0:
        (* Operation blocked (socket busy try again later) *)
    1:
        (* nothing to do here, just attempt to write again *)
        (* possibly implement a timeout and retries here *)
        (* No connection or connection has been lost *)
    -1:
        (* go back to IDLE state until there is a message again *)
        (* bad parameter to SockWrite *)
    -2:
        (* Possibilities are: unallocated socket *)
        (* or bad length *)
end_case;

```

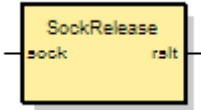
Read Data From a Socket -- SockRead()

<i>Type</i>	C Function Block
<i>Function</i>	Read data from an established connection
<i>FBD</i>	
<i>Syntax</i>	<p>SockRead(DINT SOCK, DINT MAX)</p> <p>SockRead.rslt</p> <p>SockRead.size</p> <p>SockRead.data</p>
<i>Parameters</i>	<p>sock: DINT connection to read data from.</p> <p>max: DINT maximum number of byte to read from connection.</p>
<i>Return Values</i>	<p>rslt: DINT</p> <ul style="list-style-type: none"> -2 = invalid max SIZE (must be <= 255) or invalid SOCK. -1 = no connection established. 0 = success (data available). 1 = blocked (no data available). <p>data: DINT data bytes read (STRING).</p> <p>size: DINT number of data bytes read.</p>
<i>Remarks</i>	<p><i>Read data from an established socket and place it in the DATA output string.</i></p> <p><i>SockConnect or SockListen should be called and the status checked before every read and write of the socket.</i></p> <p><i>If using SockConnect in structured text, you must declare a function block instance in the ISaGRAF dictionary.</i></p>

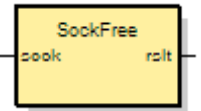
<i>ST Example</i>	<pre> (* check the connection before attempting access *) (* this check should always be done before attempting to use the connection *) (* the remote side could close the socket at any time *) SockConnectFBInst(Socket1Handle, '192.168.3.100', FALSE, FALSE); SockConnectResult := SockConnectFBInst.RSLT; if SockConnectResult = 2 then return; end_if; (* socket may be in process of releasing *) if SockConnectResult = -1 or SockConnectResult = -2 then (* bad return value do not use *) return; (* don't attempt to read *) end_if; (* SockReadFBInst is a function block instance of SockRead defined in the dictionary *) SockReadFBInst(Socket1Handle, 255); (* Read the socket to see if there is any data available *) SockReadResult := SockReadFBInst.RSLT; case SockReadResult of (* success read the data *) 0: (* No Data, wait a bit *) 1: (* nothing to do here, just attempt to read again *) (* possibly implement a timeout and retries here *) (* No Client Connection or Disconnect... *) -1: return; (* SockRead has a bad parameter *) -2: return; end_case; SockSizeResult := SockReadFBInst.SIZE; ReceiveBuffer := SockReadFBInst.DATA; </pre>
-------------------	---

Release a Socket -- SockRelease()

<i>Type</i>	C Function
<i>Function</i>	Release remote connections

<i>FBD</i>	
<i>Syntax</i>	DINT SockRelease(DINT Sock);
<i>Parameters</i>	sock: DINT Socket handle to be released from remote connection.
<i>Return Values</i>	rslt: DINT -2 invalid SOCK. 0 = release complete. 1 = in progress (retry call later until complete).
<i>Remarks</i>	None
<i>ST Example</i>	<pre> (* Release the connection to remote client *) SockReleaseResult := SockRelease(Socket1Handle); case SockReleaseResult of (* Success *) 0: return; (* Operation blocked (socket busy try again later) *) 1: (* nothing to do here, just attempt to write again *) (* possibly implement timeouts and retries here *) end_case; </pre>

Free A Socket From Memory -- SockFree()

<i>Type</i>	C Function
<i>Function</i>	Release remote connections and free socket link handle.
<i>FBD</i>	
<i>Syntax</i>	DINT SockFree(DINT SOCK);
<i>Parameters</i>	sock: DINT Socket handle to be released from remote connection.
<i>Return Values</i>	rslt: DINT -2 invalid SOCK. 0 = release complete. 1 = in progress (retry call later until complete).
<i>Remarks</i>	None

<i>ST Example</i>	<pre>(* Release the connection to remote client and free the handle *) SockFreeResult := SockFree(Socket1Handle); case SockFreeResult of (* Success *) 0: return; (* Operation blocked (socket busy try again later) *) 1: (* nothing to do here, just attempt to write again *) (* possibly implement timeouts and retries here *) end_case;</pre>
-------------------	--

ISaGRAF HiBeam Web HMI

HiBeam is an ISaGRAF tool provided in ScadaWorks to build Java Applet pages that can be display and allow users to interact with register data. HiBeam is tightly integrated with the ISaGRAF 5 application database and has access to all registers in a given project. HiBeam pages are served from the Pinnacle and later controller's Web Interface and can be accesses from any browser with a Java Plug-in installed.

In This Section

Getting Started	639
HiBeam Registration	639
Starting a HiBeam Project	640
Setting Up Accounts.....	640
Setting Up a HiBeam Screen Builder Project.....	641
Building Our First Screen	644
Downloading Our Screens	645
Accessing Our New Page From a Web Browser	647

Getting Started

Install the ISaGRAF License Manager.

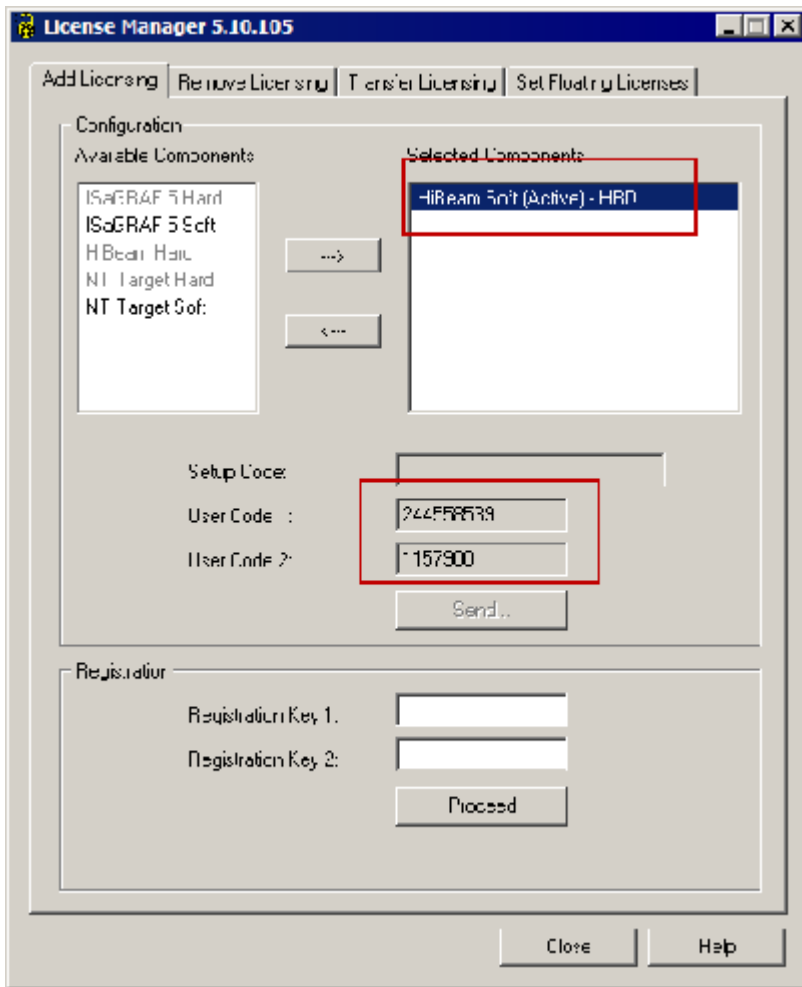
Install the HiBeam Software (available for download or on your ScadaWorks CD). This will give a 30 day demo package which can be registered at any time.

In most cases, these packages are installed by default with the ScadaWorks Installer Utility software package.

HiBeam Registration

Run the ISaGRAF License Manager. It is most often located at Start | Programs | ICS Triplex Products | "Licensing ISaGRAF 5". It is sometime s installed in Start | Programs | ISaGRAF Xpress.

You should see a window like this:



Move the HiBeam Software license over to the “selected components” window. Select the “unlimited” license in the prompt.

You will need to send ICL support (support@iclinks.com) the two “User” numbers. It may take several days to get numbers back as they must come directly from ISaGRAF. You will get back the Registration Key 1 and Registration Key 2. Once entered, you will be licensed.

Starting a HiBeam Project

Creating a new project using ScadaWorks (see Pinnacle Quick Start Guide). Make a note of where the project resides as you will need to know this to reference the HiBeam project to your application. Have the host node ready and connected with Ethernet and FTP enabled as the Pinnacle Quick Start Guide instructs you to do. This is critical for setting up a HiBeam hosting controller.

Setting Up Accounts

Open the **Node | Settings** dialog and click on the **ISaGRAF/FTP/HTTP/Admin** tab. Here we need to do two things, setup an FTP account for transferring files and turn on the HTTP server.

Make sure that FTP is checked and click the Passwords button.

User ID	Password	Directory	RD	WR	MD	CD
hibeam	admin	WebRoot	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Add an account that has the Initial Directory of WebRoot (this is case sensitive). Make sure that all permissions are on.

Enable HTTP by checking its box and click on the Permissions button.

User ID	Password	Directory	Page	G	P	C
hibeam	web			<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

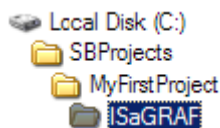
Enter a new account (this will secure your web pages and require a login. Don't forget to click the Add button.

Click the Apply to Target button and Restart Target or do a **Target / Send Complete Controller Setup...** (see "Node Configuration - Downloading" on page 30) to have the change take effect.

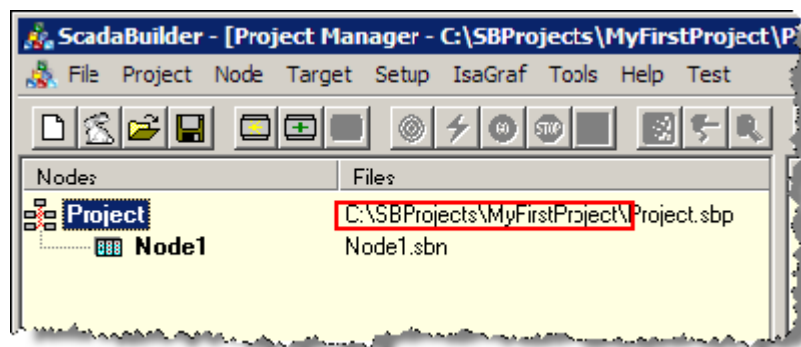
Setting Up a HiBeam Screen Builder Project

Open the HiBeam Screen Builder located at Start | Programs | ICS Triplex Products | HiBeam Screen Builder x.x | HiBeam Screen Builder. (your location may vary slightly from this).

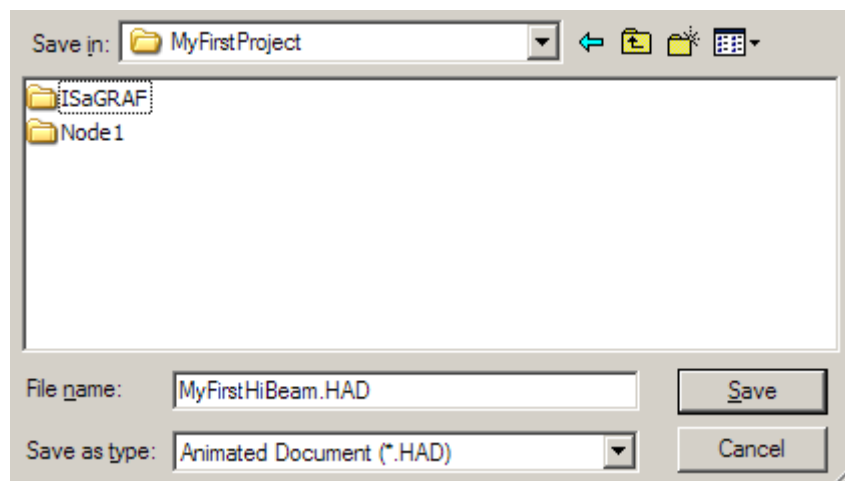
Select the File | New menu and browse to the ISaGRAF project directory within the ScadaBuilder Project. If you followed the Pinnacle Quick Start Guide to the letter, it would be located at this path:



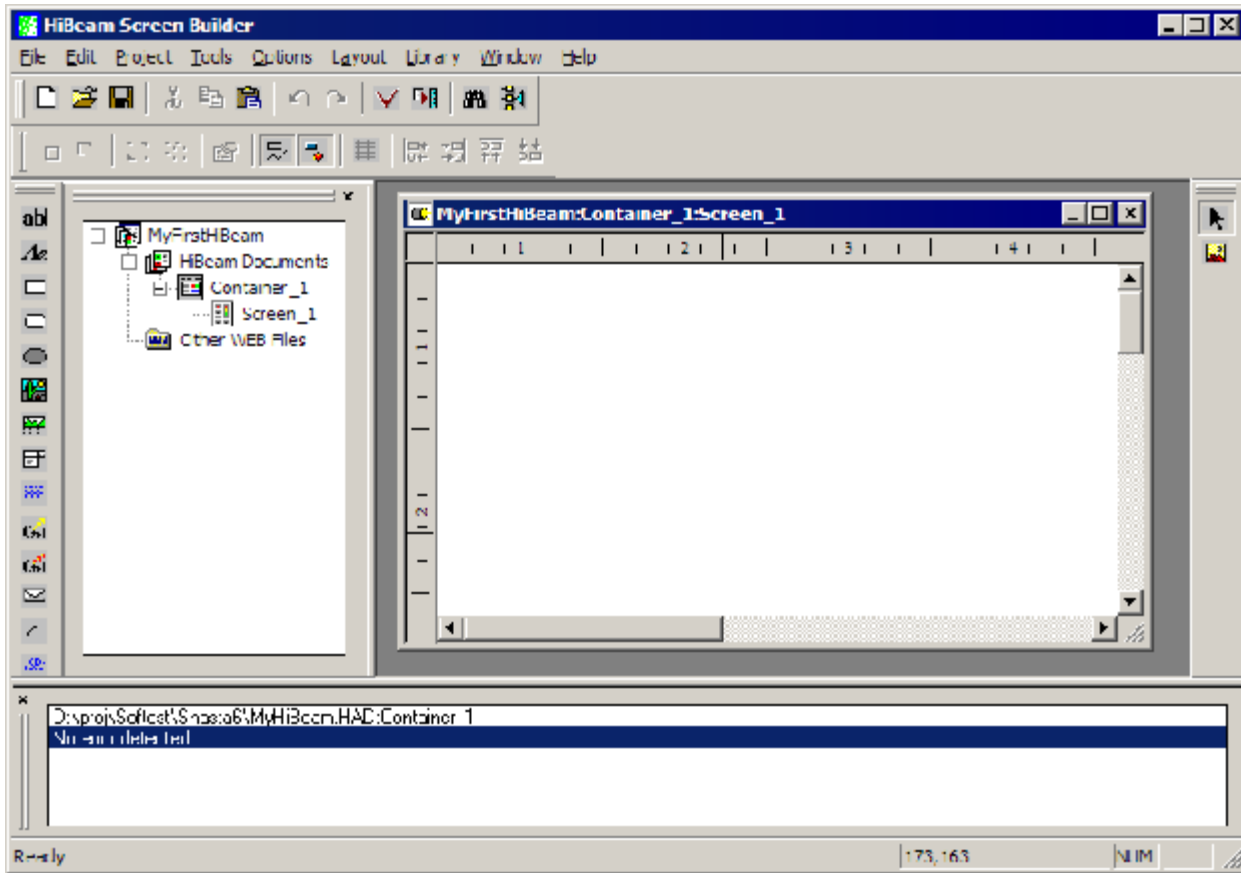
You can find the path of your project in the ScadaBuilder Workbench here:



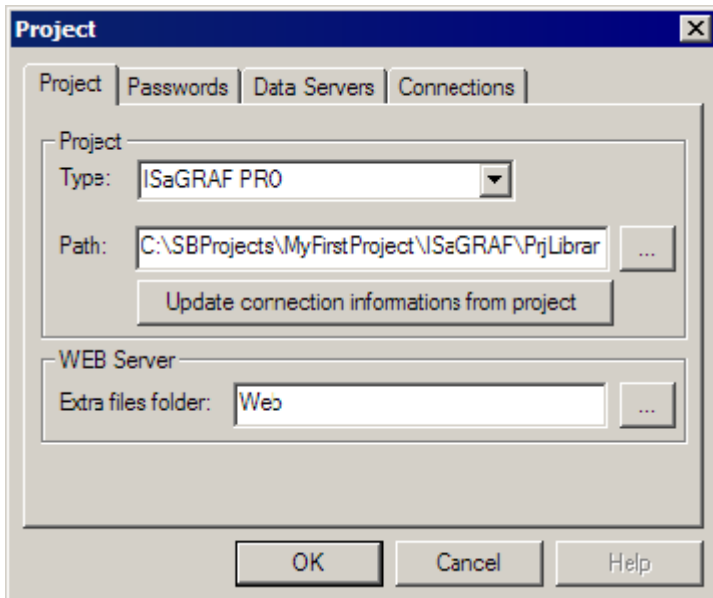
Your HiBeam Project should be located in the root of the ScadaBuilder project tree. Give the project a name and leave the .HAD extension on the file name like so:



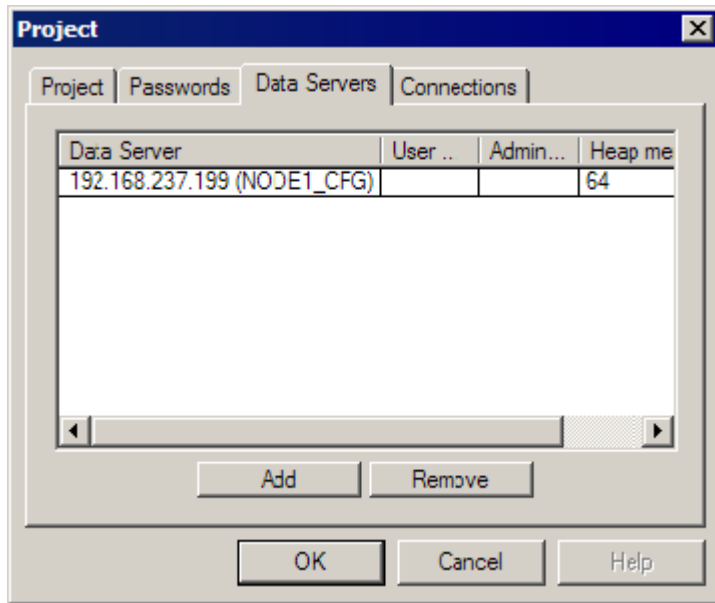
You should end up with a blank project in the HiBeam Screen builder:



Next, we need to tie this HiBeam project with its ISaGRAF project. To do this, select the Project | Settings menu. Browse to the ISaGRAF directory within the ScadaBuilder project directory shown above:




The path chosen here is: C:\SBProjects\MyFirstProject\ISaGRAF\PrjLibrary.mdb. Click on the Update connection informations from project button. Check the Data Servers tab to verify the IP address and the NodeCfg applied to the target:

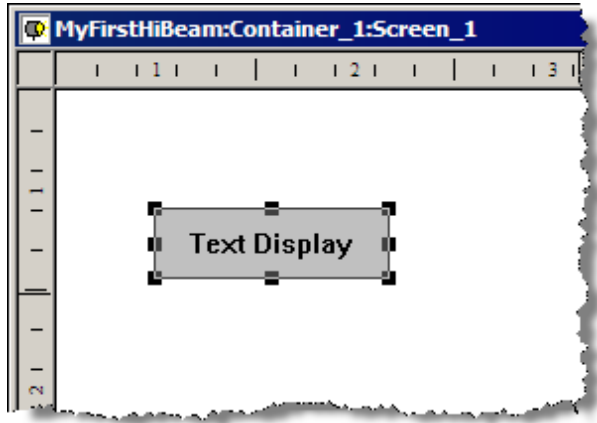


This connection will automatically import the tag database for that application. We can start building screens.

Building Our First Screen

Select the Button  tool on the left of the HiBeam Screen Builder:

Immediately drag a rectangle in the blank form page:



You can size this to your liking after inserting it as well. Right click on the control and select Properties.

There are many configurations here; we are interested in initializing the variable that this control will be tied to. Click on the Variables tab and click the ellipsis:

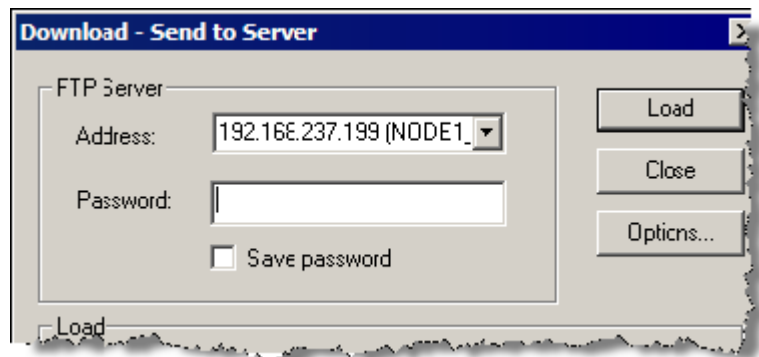
Parameter	Variable	Origin
Input Text (Input)		2: NODE1_RSC (NODE1_CFG)

Browse to the cold_junction_temp variable and select it. Click OK.

Downloading Our Screens

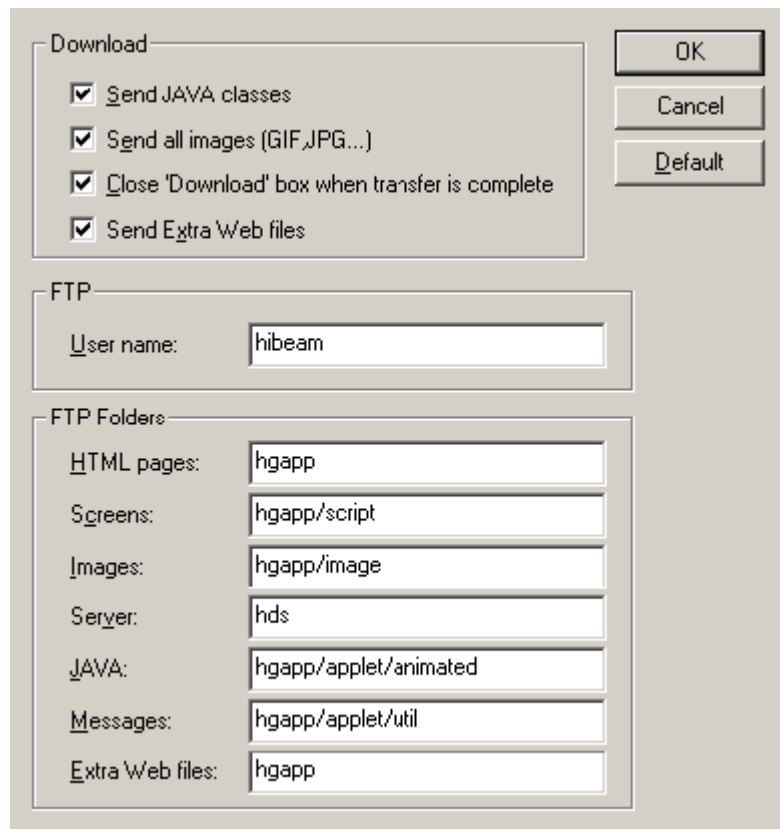
Click on the Download button  or select the Project | Send to Server... menu.

Select your address that concurs with your controller:



Click on the Options... button.

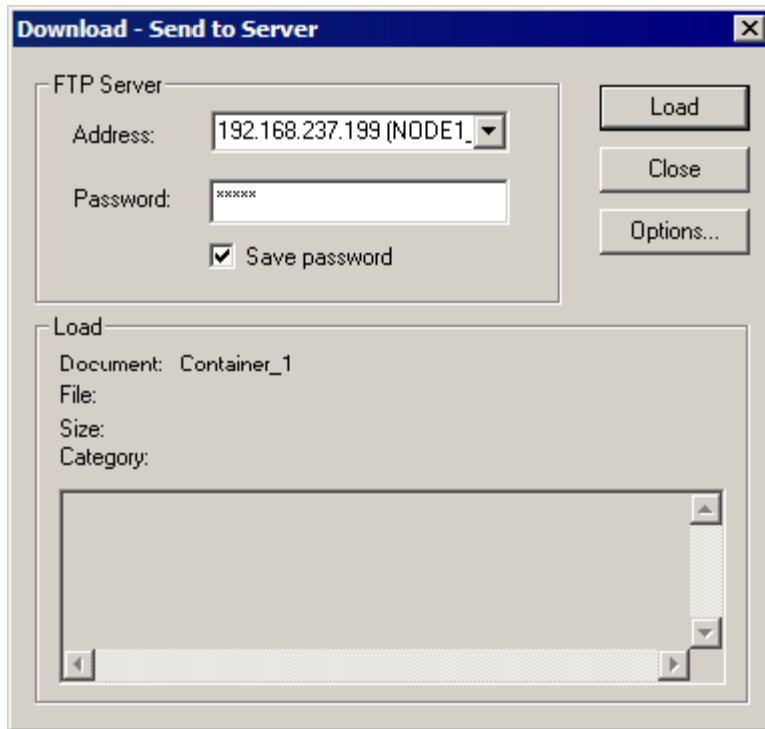
Make sure that your options are configured like this for the first download. The User name must match the FTP user name we entered earlier. All the check boxes will be checked for the first download. After the first download, you can uncheck the Send JAVA classes to speed up the download. Each unit must have the JAVA classes on it for HiBeam to function.



Note: to speed up future HiBeam application downloads, you may uncheck the Send Java Classes option above.

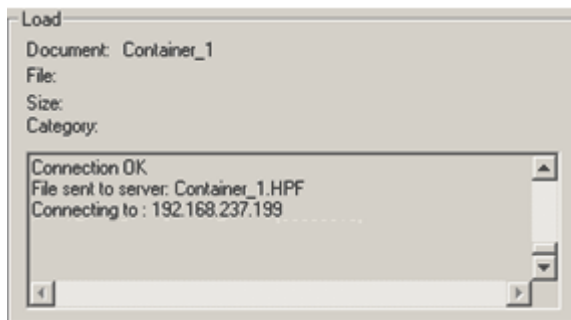
Click OK

Enter the password as configured in the FTP account earlier in the document (the password admin is shown here):



Click Load.

You will see a significant number of files download to the unit. This is normal. It will end on an error but the error is harmless and will not affect operation:

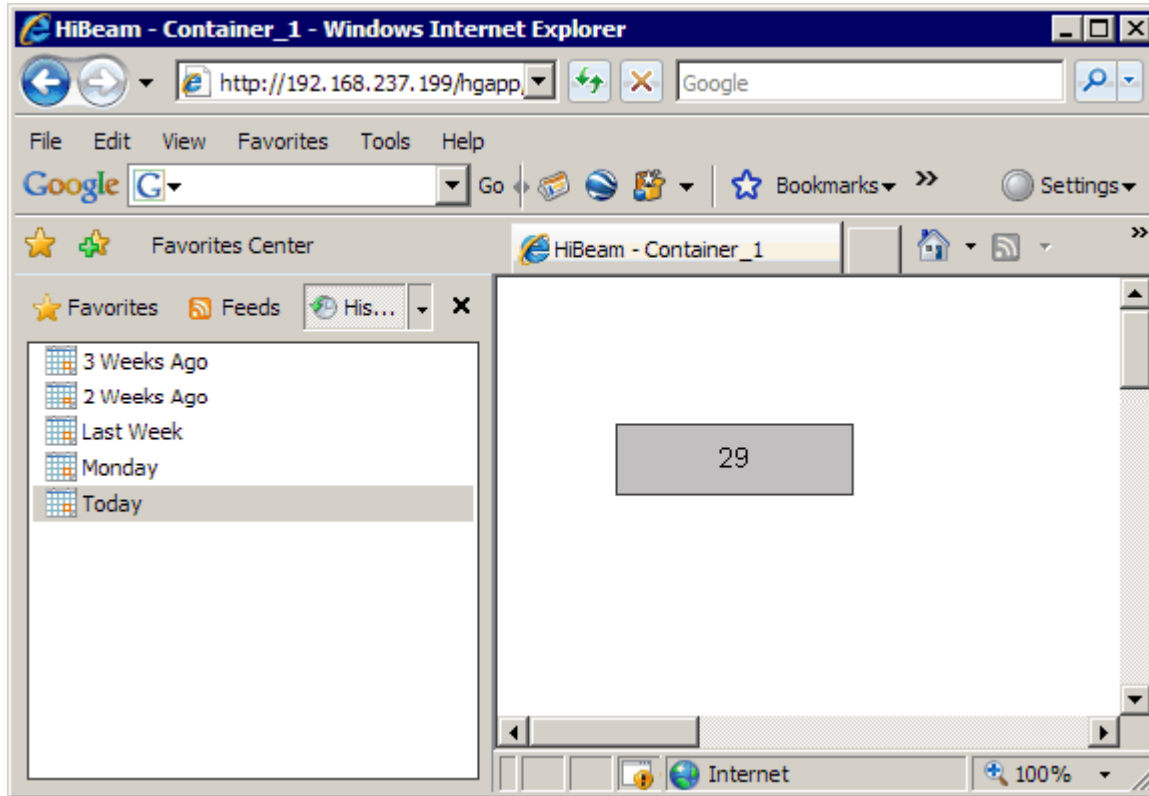


Accessing Our New Page From a Web Browser

Your application will now be accessible from your browser. Enter the path:

http://192.168.237.199/hgapp/container_1.html (substitute your IP address if necessary). The browser will ask you for a user name and password. Use the one we configured earlier: Username = hibeam and Password = web .

After entering the user name and password, you should see the screen that you have built with the temperature in integer form in your browser like so:



Feel free to start adding controls to your screen, and then more screens! You can even insert your own graphics!

Appendix A, An Ethernet/Internet Primer for TCP/IP

Confused about Ethernet and TCP/IP? Here's a quick tutorial to help...

TCP/IP Addressing

TCP/IP is a routable networking protocol in use over many computer and controller networks—including the Internet. A TCP/IP address is like any other network address. It identifies a Node on a Network individually from all other Nodes.

A TCP/IP address is a 32 bit number divided into four, 8 bit numbers. Some example TCP/IP addresses are:

- 10.1.1.32
- 172.1.114.1
- 192.168.1.199

TCP/IP addressing is simple; every device on the network has its own address and may be accessed from any other device on the network. The difficulty in understanding most networks comes from dealing with Subnets.

Networks are configured in the following hierarchy:

Network (such as the Internet)

Subnet1

Node1

Node2

Node ...

Subnet2

Subnet(x) ...

Subnets are a way of routing different messages to different networks. There are far more TCP/IP devices in the world than can be addressed by one 32 bit number (which is somewhere around 4 billion nodes or 2^{32}), so the routing capability of TCP/IP is used to create subnets.

Subnet masks are used to tell whether a message needs to use a Gateway or not. Each Subnet has a Gateway that allows nodes on one subnet to access nodes on another subnet.



Gateways are also referred to as “routers” and in some cases “firewalls”.

Let's look at an example of a TCP/IP address and subnet mask:

In this example, 192.168.1.199 is the TCP/IP address of the controller. The “Network Mask” or subnet mask of the unit configures when it will use the configured gateway (router) to get to another node or computer on another subnet. In this case, all addresses (computers, nodes, and controllers) using 192.168.1.x may communicate with one another on the local subnet. An address of 192.168.2.1 uses a different subnet will be routed out the gateway and responses will return via that same gateway.

Specify IP address:

Node IP address (LAN): 192 . 168 . 1 . 199 ☐ Read address segment from switch.

Network mask: 255 . 255 . 255 . 0

Network gateway: 192 . 168 . 1 . 254

Internet IP address (WAN):

The Network Mask says that all bits that are on define numbers of different subnets. All bits that are off define numbers of the local nodes. A network mask of 255.255.255.0 says means that y.y.y.x defines the first three “y” bytes as the subnet numbers and the x byte as the number of local nodes. For 255.255.255.0 there are 224 or 16777216 subnets and there are 256 local nodes or computers.

There are also several default classes of addressing under TCP/IP. TCP/IP addressing, including the A, B, and C classes of TCP/IP addresses and their default subnet mask numbers are shown here:

- Class A – the default subnet mask is 255.0.0.0; TCP/IP address range 1.x.x.x to 126.x.x.x
- Class B – the default subnet mask is 255. 255.0.0; TCP/IP address range 128.x.x.x to 191.x.x.x
- Class C – the default subnet mask is 255. 255. 255.0; TCP/IP address range 192.x.x.x to 223.x.x.x

Private Networks

Most “private” networks are set up to use a function of routers (gateways) called Network Address Translation. The router (gateway) uses a predefined set of addresses for the private network and one TCP/IP address for a connecting to an external network, like the Internet. The router (possibly called a firewall) will keep track of outgoing messages and the destination of the messages. When the response messages return, the router will know what private address the packet is destined for and substitute it in the message. This mechanism allows private networks to operate behind a firewall (router) with many computers and controllers sharing one “public” TCP/IP address. This mechanism also saves on static IP addresses (these addresses are individual throughout the Internet) and the cost associated with them. Firewall routers also provide the local network with protection from hackers working over the Internet because firewalls can identify unsolicited packets (TCP/IP messages).

In many cases, ICL controllers will be placed on a private network. Communicating to other controllers on the same subnet is easy. Communicating to the outside world requires that the subnet mask and the gateway be set up appropriately so that the outside gateway will “route” the message properly and that the controller will ask to be routed properly when accessing the public network or Internet.

Routing

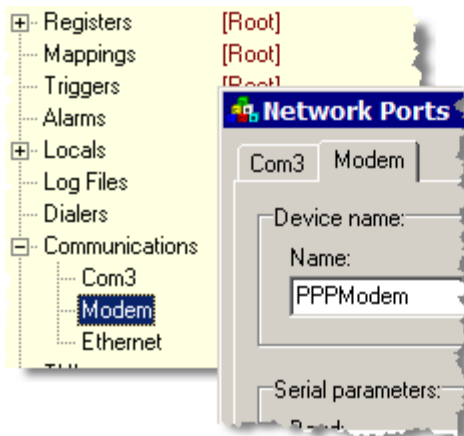
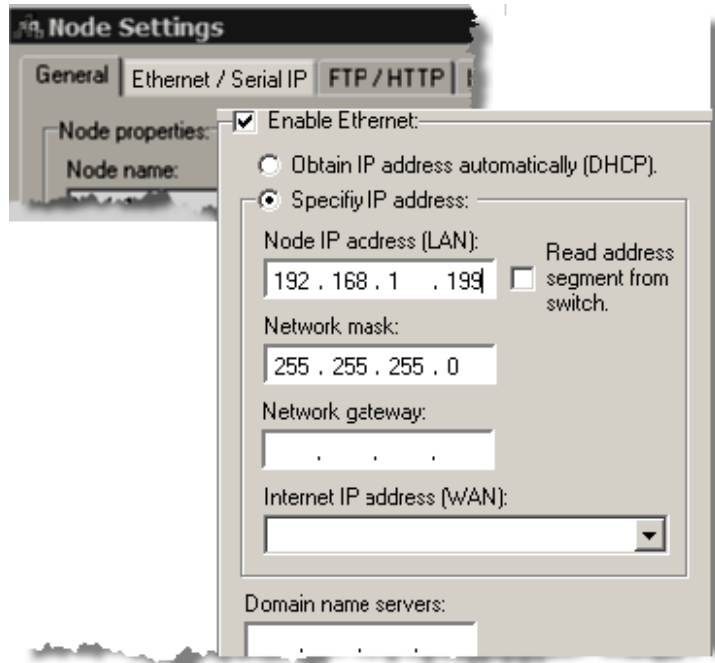
The ICL controllers are also capable of limited routing out of necessity. For example, the EtherLogic LC may have an Ethernet port for talking over a local network and a Modem (Dialup) port configured for talking to an Internet Service Provider or acting as a PPP (Point-to Point Protocol) server.

When using two different interfaces on the same controller, each interface must be configured to operate on a different subnet. To the controller, all TCP/IP operations really operate from the same “stack” or software interface and messages are “routed” to the appropriate hardware interface using the destination TCP/IP address and the routing table. Let's use an example where the EtherLogic LC has a local Ethernet interface and is using a dialup interface so that users can call in to download log files via FTP from the controller.

Now let's look at a practical example. Create a Quick Project and configure a unit as an EtherLogic LC with Modem option.

Click the Ethernet / Serial IP tab and configure the following:

The controller is configured for TCP/IP with an address of 192.168.1.199 and a Network Mask of 255.255.255.0. The Network Gateway does not matter since we are not accessing a remote network or the Internet over this interface. This dialog configures the controller to turn on TCP/IP functionality and configures the defaults for the Ethernet port as the primary interface.



Clicking OK here we can go to the Communications branch of the Setup tree in ScadaBuilder and bring up the Modem Port dialog by double clicking.

Setup the port baud rate and other parameters for your system. See *Using a Dialup Network Port* (on page 202) for more information.

Click on the TCP/IP button in the lower right hand corner.

Check “Enable Serial TCP/IP”. Select “PPP” for the interface class and check Enable server services.

You are configuring the TCP/IP identity of the controller and the remote client computer that is dialing into this port.



This is a separate TCP/IP interface from the one configured on the Ethernet port.

Configure the Server IP address to 192.168.2.1. This is the local IP address of the controller over this interface.

Configure the IP Address (remote) to 192.168.2.2. This is the remote address the controller will provide when the PPP login is successful.

Enter a user name and login. When setting up a PPP connection to the controller (like through Windows Dialup Networking), this is the username and password that you will provide to login to the controller PPP “server”.

Your dialog should look like the figure above.

Notice that the IP address here is a different subnet than specified in the Node | Settings Ethernet Serial / IP tab. This is required for routing messages to the proper port.

Click OK and then OK again in the Network Ports dialog to accept the changes.

Double Click on the node (or use the Node | Settings menu) to get to the Node Settings dialog again.

Move to the TCP/IP tab and click on the Routing button.

Routing default interface:

Network port:

Routing entries:

Network port: Destination (node, IP addr/mask or host name):

Network port	Destination	Type
PPPModem	216.171.214.0	IP address/mask

Internal routing entries:

Network port	Destination	Session
PPPModem	192.168.2.2	[server]

The TCP/IP Routing dialog also allows you to specify what TCP/IP addresses are used on what interface, and to specify the default interface.

In our case, we want the default interface to be the Ethernet port. Leaving the (default) option in the Network port assumes that TCP/IP is enabled and an Ethernet port is configured. ScadaBuilder automatically establishes the routes that use the PPPModem port if a server or a Network Session is going to use it.

If you wanted to add a route that was independent of the Network Session interface, you would add it here.

In the routing section of the frame, enter the information shown and click the Add button. You should see the route added to the route list.

This configuration sets up the PPPModem port to handle all IP addresses from 216.171.214.0 to 216.171.214.254 and to handle all others via the Ethernet port.

Other routes may be added for outgoing messages as well. For example, if your controller is configured for e-mail using a Dialup port, you will want to route all messages from controller to the e-mail server out the PPPModem port. The following would be a typical configuration (bear in mind that the Domain Name Servers fields in the Node Settings | Ethernet / Serial IP Tab need to be configured before names can be used for IP addresses):

The result here would be to take care of routing outgoing e-mails to the PPPModem interface and to route messages required by the incoming PPP server link we setup earlier. Postoffice.pacbell.net represents one address. 192.168.2.0 represents a whole subnet.



After changing anything in the Network Port / TCP/IP dialog, or the Ethernet / Serial IP section of the Node Settings dialog, it is necessary to do a Target update. A Target / Send Complete Controller Setup... is the most sure way of making sure all configurations have been downloaded to the controller.

Index

>

>> Button • 412

A

Abort Register • 236
 Abort Timeout • 236
 Accept Broadcasts • 239
 Access Code • 186
 Access Mode • 277
 Access Name • 186
 Accessing a Trend Web Page • 456, 467
 Accessing Alarms in Alarm Groups • 450
 Accessing Our New Page From a Web Browser • 648
 Accessing the User Portal for Administration • 426
 Account/Domain Name • 239
 Ack Attr • 401
 Ack Mode • 168
 Ack Text • 401
 Acknowledge Trigger • 163
 Acknowledged Text • 101
 Action • 255
 Activation Add Button • 261
 Activation Trigger • 163
 Active Alarms! Popup • 449, 452
 Activity Timeout • 205, 225
 Add Button • 45, 122, 127, 157, 163, 186, 413
 Adding a New User Account • 427
 Adding and Accessing Web/HMI Links Via a User Account • 434
 Adding and Editing Web/HMI Links • 432
 Address • 245
 Address Com Fail • 246
 AI Message Register Link • 225
 Alarm • 99, 167, 401
 Alarm Acknowledge String • 177
 Alarm Acknowledge Tag • 491
 Alarm Add Button • 99
 Alarm Annunciation Register • 167
 Alarm Auto Log Enable • 164
 Alarm Button • 101
 Alarm Dialer Reference • 174
 Alarm Group • 164
 Alarm List Repeat Count • 176

Alarm List Repeat Delay • 176
 Alarm List Retry Count • 238
 Alarm List Wait • 238
 Alarm Message • 174
 Alarm Options • 161, 448, 451
 Alarm Retry Count • 238
 Alarm State • 187
 Alarm State Tags • 491
 Alarm Suitable Triggers • 147, 159
 Alarm Text • 164
 Alarm Wait • 238
 Alarms • 448
 Alarms Button • 87
 Alarms Reference • 163
 Alarms Tab • 491
 Allocate a new socket -- SockAlloc() • 629
 AO Message Register Link • 226
 Appendix A, An Ethernet/Internet Primer for TCP/IP • 37, 39, 51, 369, 375, 437, 649
 Apply Button • 44, 46
 Apply Scaling Record To Analog Channel -- ioscale() • 580
 Applying a Scaling Record • 135, 140
 Archive Activation Trigger • 87
 Archive Indicator Map • 88
 Attachments • 372
 Attributes • 396, 398
 Authentication Type • 209
 Auto Answer Button • 205
 Auto Edit • 394, 403, 408
 Auto Select • 411
 Averaging Decimal Places • 462

B

Baud • 200
 Before you get started • 18
 Bit Packing and Unpacking functions • 572
 Block Size • 95, 100, 119, 135, 136, 157, 255, 275, 402, 407, 410
 Blocksize (Address) • 230
 Boolean Registers • 444
 Boolean Top Margin (%) • 461
Boolean, Integers and Real Register Windows • 104
 Border • 406
 Bricknet Protocol • 204, 348
 Buffer • 151, 403
 Buffer Format Editor • 126, 177
 Building Our First Screen • 645
 Button Attributes • 397, 399, 405

C

- Call Group • 174, 186
- Call Group Reference • 172, 175
- Cc • 372
- Cc Buffer • 372
- Cell Modem Status Map • 212
- Cell Phone Number Map • 213
- Cellular Button • 202
- Change Dir Checkbox • 44
- Change FTP Transfer Type -- FtpType() • 567
- Changing The IP Settings • 436
- Changing Your Password • 441
- Characterize A Non-Linear Instrumentation Curve -- charctrz() • 581
- Check Disk Space Or Create A Directory -- diskmgmt() • 552
- Check For End Of File Status -- f_eof() • 529
- Check To See If A File Exists -- f_exist() • 542
- Clamp • 408
- Clamp Min & Max • 140
- Clear The Receive Buffer -- ComClrRcv() • 514
- Client Connection Basic Strategy • 629
- Client Login • 209
- Client Response Timeout • 42
- Clock Speed Set / Map • 494
- Close a Communications Port -- ComClose() • 506
- Close A File -- f_close() • 541
- Close A Network Port By Handle -- NPClose() • 525
- Close An FTP Connection -- FtpClose() • 565
- Close Button • 126
- Columns • 394
- Com Disable • 246
- Com Statistics • 247, 253, 293, 305, 318, 334, 343, 352
- Com Statistics List • 247
- Com Statistics Register • 247
- Com Statistics Select • 247
- Comm Error Register • 498
- Comm Stats Checkbox • 124
- Comm Timeout • 498
- Command (Hart Supported) • 341, 345
- Command Mode • 259
- Comments • 122
- Communications** • 191
- Configuring Traces and Trace Parameters • 463
- Confirm Retries • 224
- Confirm Timeout • 224
- Confirm Writes • 490
- Connect To a Server or Initialize Socket -- SockConnect() • 630
- Connection Timeout • 225
- Control DTR On The ComPort -- ComDtr() • 516
- Control RTS On The ComPort -- ComRts() • 515
- Controller Baud Rate • 41
- Controller Debugger Port • 41
- Controller Options • 35, 203
- Controller Serial Number Map • 495
- Controller/RTU Model • 35
- Controlling VUI Access • 188
- Controls • 148, 399
- Copy A File -- f_copy() • 546
- Copy A Large File -- f_copy_l() • 548
- Copyright Notice • 2, 60, 218
- Count • 122
- Count Button • 95, 100
- Count Mode • 227
- Counter Scan Rate • 36
- Create Host File Transfers • 236
- Creating a Bricknet Destination • 350, 363
- Creating a Bricknet Interface • 282, 348, 356, 362
- Creating a DF1 Master Destination • 303
- Creating a DF1 Master Interface • 302
- Creating a DF1 Slave Interface • 308
- Creating a DNP 3 Master Destination • 316
- Creating a DNP 3 Master Interface • 315
- Creating a DNP 3 Slave Interface • 312
- Creating a Hart Master Destination • 342
- Creating a Hart Master Interface • 341
- Creating a Local Alarm** • 165
- Creating A Local Event** • 155
- Creating A Local HMI • 416
- Creating a Log Alarm** • 97
- Creating a Log Event** • 92
- Creating a Log File** • 80
- Creating a Modbus Master Destination • 252, 270
- Creating a Modbus Master Interface • 249, 270, 281
- Creating a Modbus Slave Interface • 262, 270, 281
- Creating a TUI** • 389
- Creating a Voice User Interface** • 183
- Creating An Alarm** • 160
- Creating an Alarm Dialer** • 169
- Creating an Email Interface • 217, 368

- Creating An Ethernet IP Master Destination • 333
- Creating An Ethernet IP Master Interface • 332
- Creating An Ethernet IP Slave Interface • 339
- Creating an FTP Client Interface • 217, 374, 565
- Creating an FTP Event • 78, 377
- Creating An SDX Destination • 291, 298
- Creating An SDX Session • 289
- Creating Bricknet Network Events • 267, 352, 364
- Creating DF1 Master Events • 305
- Creating DNP 3 Master Class Events • 326
- Creating DNP 3 Master Events • 320
- Creating DNP 3 Master Objects • 319
- Creating Emails • 78, 370
- Creating Ethernet IP Master Network Events • 336
- Creating Ethernet IP Network Message Links • 339
- Creating Hart Master Events • 343
- Creating Modbus Master Events • 218, 254, 267, 281
- Creating SDX Network Events • 293
- Current File Bytes Register • 235
- Current Modem Driver • 228
- Current Xfer Bytes Register • 236
- Cursor
 - Mode Hovering and Zooming With iPad and iPhone Browsers • 481
- Cursor Hover Display • 462
- Customizing the Voice Interface (English and Non-English) • 180
- Cycle Count • 149, 258

D

- Data Bytes Limit (MB) • 459
- Data Days Limit (days) • 459
- Data Days Register Map • 458
- Data File Directory • 459
- Data File Drive • 459
- Data Link Retries • 224
- Data Link Timeout • 224
- Data Mapping • 498
- Data Pack • 275
- Data Sample Rate (Sec) • 459
- DataBits • 200
- Date Button • 87, 95, 100, 175, 373
- Date Format • 83
- Date Range Button • 475
- Day Checkboxes • 151
- Daylight Savings Map • 212, 495
- Deadband Hysteresis • 152
- Debounce Time • 152
- Decimal Column • 491
- Decimal Places • 84, 95, 100, 174, 186, 372
- Decimal Width • 127, 408
- Default Monitor-Answer Session • 189, 238
- Default Session Activity Timeout • 210
- Defining a Local Alarm • 166
- Defining A Local Event • 155
- Defining a Trigger • 144**
- Defining An Alarm • 160
- Delete A File -- f_delete() • 543
- Delete Path • 230
- Delta Mapping • 499
- Deployment • 612, 627
- Destination • 256
- Destination File • 280, 380
- Destination Index/Element • 257
- Detect When The Transmit Buffer Is Empty -- ComXmtEm() • 513
- Device To Register Mapping • 494
- DF1 Protocol • 302
- DI Message Register Link • 226
- Diagnostic Map Start Register • 206
- Diagnostic Tab • 231
- Dial Prefix • 205
- Dial Retry Count • 228
- Dialer Activation • 174
- Dialing Tab • 177, 248
- Dialup Button • 202
- Dialup Modem Parameters Reference • 205
- Dialup Tab • 228
- Directory • 45
- Disable Archive File Overwrite • 88
- Disable Attributes • 397
- Disable I/O Scan From Running On Target Controller • 49
- Disable Register • 406
- Disable Session at Startup • 239
- DNP 3 Protocol • 310
- DO Message Register Link • 226
- Documents • 453
- Domain Name Server (DNS) Retries • 42
- Domain Name Servers • 40
- Downloading Our Screens • 646
- Drive • 84

E

- Edit Button • 174
- Edit Buttons • 412
- Editing and Deleting User Accounts • 430, 441

Elap Button • 95, 100
 Email Date Format • 373
 Email Protocol • 367
 Email Reference • 372
 Enable • 205
 Enable Admin Login • 46
 Enable Client Services • 208
 Enable Compression • 84
 Enable CPU WDT • 36
 Enable Ethernet • 38
 Enable File Transferring • 240
 Enable FTP • 42
 Enable HTTP • 44
 Enable I/O Scan Sync Trigger • 36, 88
 Enable ISaGRAF • 41
 Enable ISaGRAF Auto Make • 32
 Enable ISaGRAF Strict Variable Checking • 49
 Enable Or Disable A Network Session By Name
 -- NSCtrl() • 554
 Enable Or Disable The Controller I/O Scan --
 IOCtrl() • 583
 Enable Serial TCP/IP • 208
 Enable Server Services • 209
 Ethernet IP Protocol • 332
 Ethernet Tab • 33
 Event • 258
 Event - Address • 279
 Event Class • 276
 Event Gap • 224
 Event Message Routes • 229
 Event Name • 255
 Events Button • 87
 Excel Button • 86, 95, 100

F

File Archive Enable • 87
 File Button • 123
 File Export Button • 123
 File Footer • 84
 File Header • 84
 File I/O Functions • 526
File Menu • 55
 File Mode • 280, 379
 File Name • 83, 88
 File Number • 275
 File Transfer • 234, 279, 361
 File Transfer Event Reference • 279, 361
 File Transfer Protocol (FTP) Functions • 565
 File Transferring Over Modbus And Bricknet •
 78, 218, 278, 359
 File Type • 379

Finishing the Alarm Dialer Configuration • 173
 Flip Flop Latch Function -- flipflop() • 595
 Font Color • 460
 Font Name • 460
 Font Style • 460
 Force New Header • 88
 Format Button • 125
 Free A Socket From Memory -- SockFree() •
 637
 FTP Client Status Buffer • 496
 FTP Event Reference • 379
 FTP Passwords Button • 43
 FTP Server Port Number • 42
 FTP Server Status Buffer • 496
 FTP User Passwords Editor • 43

G

Gateway (optional) • 209
 General Notes. • 612
 General Settings • 392
 General Tab • 32, 219, 490
 Generate Button • 182
 Generate Voice Reference • 175, 176, 182
 Generating Alarm and Alarm Group Reports •
 449, 451
 Generating and Downloading a Voice File • 179,
 185
 Get A File Over FTP -- FtpGet() • 566
 Get A Netport Handle By Name -- NPHandle() •
 521
 Get Files Checkbox • 46
 Get The FTP Client Status -- FtpCStat() • 567
 Get The Maximum Scan Time -- scanmax() •
 605
 Get The Number of Bytes In The Receive Buffer
 -- ComRCnt() • 508
 Get The Number Of Milliseconds Since Startup -
 - systick() • 606
 Get The Scan Time Of The Previous Scan --
 scantime() • 605
 Getting Event Data With a SCADA System •
 330
Getting Started • 639
Getting Started with a Project • 23
Getting To Know ScadaWorks • 15
 Global Com Statistics • 231
 Global Com Statistics Add Button • 234
 Global Com Statistics List • 234
 Global Positioning Satellite (GPS) Interface • 63,
 497, 604
GPS Reference • 497

Gradient End Color • 461
 Gradient Start Color • 461
 Graph Attributes • 397, 402
 Greeting Message • 185
 Group • 397
 Group / Variation • 276
 Group Column • 491
 Group Name (Session) • 175

H

Hart Master Protocol • 341
 Header • 394
 Header Attributes • 396
 Height • 405
 HiBeam Registration • 639
 High Speed Log Enabled • 88
 History • 406
 Hop (Node Name) • 230
 Host Node • 36
 Hour/Min Constants • 151
 Hour/Minute Registers • 151
 HTML Page • 45
 HTTP Permissions Button • 44
 HTTP Permissions Editor • 44, 70
 HTTP Server Port Number • 44
 HTTP Server Status Button • 496

I

I/O Checkbox • 124
 I/O Configuration - AI AC Filter Mode • 137
 I/O Configuration - DI Gate Time • 136
 I/O Configuration - DO Flash Rate • 136
 I/O Configuration - ICL-4300 (PC-in-a-Brick) Controllers • 138
 I/O Configuration - Map Button • 135
 I/O Configuration - Scale Button • 135
 I/O Configuration - Temperature Average Time • 137
 I/O Configuration - Unmap Button • 135
 I/O Configuration - Unscale Button • 135
 I/O Configuration - Watch Dog Timer • 136
 I/O Map Reference • 134
 I/O Mode • 140
 I/O Options • 136
 I/O Range Low & High • 141
 I/O Ranges for Different Scaling Modes • 141, 142
 I/O Scale Assignment Reference • 136
 I/O Scaling • 139
 I/O Scan Cycle Multiplier • 88, 89
 I/O Scan Sync Rate • 36, 88, 89

Idle Attr • 401
 Idle Mode • 168
 Idle Text • 101, 401
 In Progress Map • 259
 In Start
 (Address) • 229
 Include domain name servers in routing table • 40
 Include Domain Name Servers In Routing Table • 40
 Include ISaGRAF Application Files • 49
 Incoming Message Count Map • 213
 Index • 121
 Index Mode • 227
 Info Button • 477
 Initial Directory • 43
 Initialization String • 205
 Instrumentation Functions • 575
 Integer and Real Registers • 445
 Integer Cast Type • 227, 277
 Integer Width • 127, 408
 Interface Class • 208
 Internet IP Address (WAN) • 36
 Introduction Message • 176
 IP Address • 245
 IP Mode • 403
 ISaGRAF 3 Program Environment • 29
 ISaGRAF 3 Register Tools • 116
 ISaGRAF 5 Programming Environment • 29
ISaGRAF And ScadaWorks • 27
ISaGRAF Data Types and Function Prototypes • 501
ISaGRAF Function and Function Blocks • 501
ISaGRAF HiBeam Web HMI • 639
 ISaGRAF Menu • 64
 ISaGRAF Socket Functions (UDP and TCP/IP) • 629

K

Key Code Register • 404
 Key Prompt • 405
 Key Timeout • 405

L

Label • 401, 403
 Labels • 402, 407, 410, 412
 Language • 182
 Last Caller Number Map • 213
 Lead Delay • 200
 Left Axis Range (min, max) • 460

Legend Display • 462
 Limit Rise And Fall Rate Of An Analog Value --
 ratelim() • 586
 Link Attributes • 396
 List • 414
 List Retry Count • 178
 List Wait • 178
 Live View -- View and Edit Registers • 442
 Load (Modem Driver) • 229
 Local Alarms Reference • 167
 Local Events Reference • 157
Local HMI Navigation • 416
 Local HMI Numeric Entry • 421
 Local Side Port (PC side) • 42
 Log Alarms Reference • 99
 Log Buffer Size • 83
 Log Cycle Start Map • 88, 89
 Log Delimiter • 94, 99
 Log Duration (Sec) • 88, 89
 Log Events Reference • 94
 Log File • 405
 Log File Archive Tab • 87
 Log File Name Map • 89
 Log File Setup Tab • 83
 Log Files Reference • 82
 Log Message • 94, 99
 Log Mode • 373
 Log Trigger Count Max • 88, 89
 Logging Functions • 596
Logging In To The TMI • 485
 Logical Functions • 595
 Login Add • 188
 Login Password • 188, 237
 Login Password Buffer • 189
 Login Read Only • 189
 Login Tab • 237
 Login Tab Reference • 188
 Login User Name • 189, 237
 Login User Name Buffer • 189
Low Level Communications Functions •
 504
 Low Level I/O Port Access Functions • 569

M

Make Dir Checkbox • 44
 Managing and Editing Registers • 106
 Manually Trigger A Network Event --
 nettrigger() • 561
 Map to NVRAM (Make Retained) • 121
 Mapping and Unmapping I/O to Registers • 131
 Mappings Reference • 63, 212, 268, 301, 456,
 493
Master Protocol List • 195, 241, 250, 262,
 290
 Max File Size • 83, 87
 Max Records Count • 87
 Media Access Delay • 201
 Media Ready Mode • 201
 Memory Available Map • 494
 Message Body • 372
 Message Count Mode • 277
 Message Data Packing • 259
 Message Index Mode • 277
 Message Index or Element • 276
 Message Registers • 446
 Message Type • 275
 Messages Size • 122
 Middle Enum • 121
 Misc Tab Reference • 189
 Miscellaneous Tab • 234, 238
 Modbus Protocol • 204, 249
 Mode • 393, 408
 Mode (Strip/History Selector) • 475, 479
 Modem Driver List • 229
 Modem Port • 176

N

Name • 200, 393, 397, 401, 402, 403, 404, 405,
 407, 410
 Name List • 123
 NetPorts Button • 52
 NetSessions Button • 52
 Network Address • 151, 223
 Network Address Reference • 239
 Network Address Register • 239
 Network Control Functions • 554
 Network Destination Reference • 245
 Network Event Activation • 255, 260, 294, 307,
 322, 337, 345, 354
 Network Event and Network Message Link
 Display • 263, 274
 Network Events Checkbox • 124
 Network Events Reference • 216, 255, 294, 354
 Network Gateway • 40
 Network Mask • 39
 Network Message Link Reference • 217, 275,
 309, 313
 Network Port • 185, 223, 490, 498
 Network Ports Reference • 199
 Network Session • 151
 Network Session (Route) • 230

- Network Session Name • 258
- Network Sessions Button • 202
- Network Sessions Reference • 219, 250, 251, 289, 291, 303, 316, 342
- New Button • 174
- New Path • 230
- Node Address • 35
- Node Configuration - Downloading • 30, 425, 641
- Node File Transfer Mode • 37
- Node IP Address • 39
- NODE Menu • 56, 454
- Node Name • 34
- Node Settings - Advanced Tab • 48, 57, 583
- Node Settings - Ethernet Tab • 37, 57, 192, 199, 207, 209, 223, 239, 250, 368, 369, 374, 375
- Node Settings - FTP/HTTP Tab • 46, 57, 59, 65, 374
- Node Settings - General Tab • 34, 57, 88, 176
- Node Settings - ISaGRAF / FTP / HTTP / Admin Tab • 41
- Node Settings Reference • 26, 34
- Number of Rings Before Answering • 189
- Number Retry Count • 178
- Number Wait • 178
- Number/Buffer • 248
- Numeric and Alpha Numeric Pager Sessions • 366

O

- Obtain DNS Servers • 209
- Obtain/Specify IP address • 38
- On/Off Time • 167
- Open A Binary File For Read Only Access -- `f_ropen()` • 527
- Open A Binary File In Read-Write Mode -- `f_wopen()` • 528
- Open a Communications Port -- `ComOpen()` • 505
- Open a File For Appending -- `f_aopen()` • 526
- Open a Listening Socket To Act As Server -- `SockListen()` • 631
- Open A Network Port By Name -- `NPOpen()` • 519
- Open An FTP Connection -- `FtpOpen()` • 565
- Open File Buffer • 235
- Operation • 279, 379
- Option Trigger on Startup • 163
- Options • 127, 177
- Out Start (Address) • 229

- Outgoing Message Count Map • 213
- Override Project's PC Port Settings • 36

P

- Pack 16 Booleans Into An Integer Register -- `pack16()` • 572
- Packet Size • 236
- Page • 406
- Page Heading • 460
- Page Link Attributes • 399
- Page Position (x,y) • 459
- Page Settings • 397
- Page Size Pixels (x,y) • 459
- Page Theme • 460
- Parity • 200
- Password • 43, 45, 492
- Path Navigate • 230
- Periodically Totalize An Analog Value -- `Totalize()` • 584
- Phone Number • 177, 228
- Phone Number Buffer • 177, 228
- PID Closed Loop Control -- ICLPID • 575, 587, 590
- PID Closed Loop Control -- `PID_AL()` • 587
- Play Button • 182
- Port • 393
- Port (2nd) • 393
- Port Number • 379
- Port Tab • 30, 33, 36, 37
- Post Files Checkbox • 46
- PPP Connect Timeout • 208
- Prefix • 119
- Prefix Enum • 119
- Preview Button • 175, 176, 185
- Probe Interval • 225, 298
- Probe Interval Disable • 225, 564
- Program Implementation • 609, 623
- PROJECT Menu • 55
- Project Name • 32
- Project Settings Reference** • 32, 56
- Prompt • 397, 404
- Prompt Repeat Time • 185
- Prompt Retry Count • 185
- Protocol • 175, 219
- Protocol Status Buffer • 239

Q

- Quit Enable • 395

R

- Range • 402

- Read A 16bit Word From I/O Space -- InWord() • 570
- Read A Boolean Register By Index -- BooRd() • 592
- Read A Byte Out Of The Receive Buffer -- ComRdBt() • 509
- Read A Line From a Text File to a Message Register -- fm_readCrLf() • 531, 537
- Read A Message Register (STRING) From A File -- fm_read() • 533, 539
- Read A Message Register By Index -- MsgRd() • 593
- Read A Real Register By Index -- RealRd() • 592
- Read A Real Value From A File -- fr_read(); • 540
- Read An Integer Register By Index -- AnaRd() • 591
- Read An Integer Register From A File -- fa_read() • 536
- Read Checkbox • 44
- Read CTS From The ComPort -- ComCts() • 517
- Read Data From a Socket -- SockRead() • 635
- Read Date, Time Or Day Of Week -- day_time • 604
- Read DCD From The ComPort -- ComDcd() • 518
- Read Multiple Bytes Out Of The Receive Buffer -- ComRdBt() • 509
- Read Network Event State -- npending() • 563
- Read One Byte From I/O Space -- InByte() • 569
- Read Only • 403, 408, 411
- Read Only (Setup Tab) • 87
- Read Only Password Buffer • 186
- Read Only Password String • 186
- Read The Current RTC Seconds Since 00 00 01/01/70 -- RTCSecs • 603
- Read The Eight Position DIP Switch Value -- Read_sw() • 569
- Read The RTC Into Integers -- DateRd() • 602
- Read The Time And Date From GPS -- gpsrd() • 603
- Read/Write Password Buffer • 185
- Read/Write Password String • 185
- Read-Only Checkbox Column • 491
- Real Time Clock (RTC) • 151
- Real Time Clock (RTC) Functions • 600
- Real Time Clock Network Events • 254, 267, 354
- Reboot Map** • 494
- Receive Character Timeout • 224
- Receiver Quiet Time • 201
- Redial Retry Count • 177
- Redial Wait • 177, 228
- Redundancy Function Block For Legacy Controllers • 607
- Redundancy Function Block for Pinnacle Controllers • 614
- Redundant Function Block Parameters • 616
- Redundnt Function Block • 607
- Refresh • 394
- Refresh Button • 478
- Regenerate Default I/O Scaling Entries • 35
- Register • 84, 94, 99, 127, 174, 186, 372
- Register Block Size • 150
- Register Button • 86, 95, 100, 174, 372
- Register Column • 491
- Register List • 412
- Register List Button • 123
- Register Min & Max • 140
- Register Mode • 226, 258, 276
- Register Name • 121
- Registers • 63, 103, 362
- Registers Reference • 118
- Registers Tab • 225, 261, 272, 491
- Reinit Period • 205
- Release a Socket -- SockRelease() • 636
- Remote Host Address • 257
- Remote Host File Transfers • 36, 67, 236, 279, 281, 361
- Remote Node • 245
- Remote Node Name • 257
- Remote Scale Checkbox • 124
- Remote Scaling • 237
- Remote Scaling List • 237
- Remote Scaling Register • 237
- Rename A File -- f_rename() • 545
- Rename Path • 230
- Reports Button • 468
- Required GPS Messages • 499
- Resend Retry Count • 224
- Reset Button • 202
- Reset Communication Port Parameters -- ComSet() • 507
- Response Buffer • 206
- Response Delay • 201
- Response Timeout • 224
- Restart Remote • 280
- Retained (Non-Volatile) Registers and Initial Values • 113
- Retained Check Box • 122
- Retained Register Drop Down List • 113, 124

- Retrieve A Network Destination Handle --
ndhandle() • 559
- Retrieve A Network Event Handle -- nehandle()
• 559, 560
- Retrieve A Network Session Handle --
nshandle() • 557
- Retrieve NetEvent, NetDest, NetSession Handles
-- nethandles() • 555
- Retrigger Time • 151
- Right Axis Range (min, max) • 461
- Ring go away timer • 205
- Routing Button • 40, 50, 209
- Routing Editor - Default Network Port • 52
- Routing Editor - Entry Destination • 52
- Routing Editor - Entry Destination Type • 52
- Routing Editor - Entry Network Port • 52
- Routing Editor - Internal Routing Entries • 52
- Routing Tab • 229, 248
- Rows • 394
- RS-485 Default • 201
- RTC Retrigger Options • 153
- RTC Sync Enable • 211, 212
- RTC Sync Period • 498
- RTC Sync Period (hrs) • 211, 212
- RTS Control • 200

S

- Sample Output • 121
- Save Project to Target on Exit • 32
- ScadaWorks Development Cycle • 21**
- Scale A Linear Analog Device -- Scaler() • 583
- Scaling Entry • 136
- Scaling Tab • 237
- Scan Rate • 394
- Scheme • 395
- Secure Data Exchange (SDX and STM)
Protocols • 288, 298
- Security Level • 395, 398, 404, 405, 406, 407,
409, 410
- Security Register • 395, 398, 404, 406, 407, 409,
410
- Seek Or Get The Position Of A File -- f_seek() •
535
- Selection Attributes • 396
- Send A File Over FTP -- FtpSend() • 566
- Send A Packet Over A Network Port --
NPPktSnd() • 523
- Send Button • 182
- Send ScadaBuilder Configuration Over
Debugger • 49
- Serial Number • 37
- Server • 223
- Server Implementation Basic strategy • 629
- Server IP Address • 209
- Session Gap • 224
- Setpoint Triggers • 145
- Setting Radio Diagnostic Mode • 438
- Setting Up a Bricknet Routing System • 355
- Setting Up a Call Group • 171
- Setting Up a HiBeam Screen Builder Project •
642
- Setting Up a Modbus Routing System • 270
- Setting Up a New Trend • 63, 456
- Setting Up Accounts • 641
- Setting Up an Administrator Account •
424**
- Setting Up An SDX Routing Example • 298
- Setting Up An STM Interface • 211, 301
- Settings Button • 175, 176, 185, 472
- Setup And Technical Considerations • 614, 627
- SETUP Menu • 63
- Setup Tab • 175, 245
- Setup Tab Reference • 185
- Show Attribute Columns • 126
- Show Bank Name and Indexes • 412
- Show Internal Com Ports in the Network Ports
Dialog Window • 48
- Show Register Names • 412
- Signal Poll Period (Sec) • 212
- Signal Strength Map • 212
- Slave Configuration To Retrieve Event Data •
329
- Slave Protocol List • 197, 217, 275
- Source • 255
- Source File • 280, 379
- Source Index/Element • 256
- Special Trigger Types • 147
- Start Register • 135, 150, 157, 275, 402, 407,
410
- Starting a HiBeam Project • 640
- Startup Batch File Command Lines • 48
- State • 150
- State Button • 101
- State Map • 164
- State Register • 235
- Statistics Average Display • 462
- Statistics Range Display • 462
- Stats Button • 478
- Status Buffer • 176, 185
- Status LED Map • 494
- Steps • 248
- Stop Bits • 200

Strip Chart Duration (Hours) • 460
 Strip Chart Mode (Enable) • 460
 Subject • 372
 Suffix • 121
 Suffix Enum • 121
 Supported DNP 3 Master Commands • 328
 Switch To VUI • 176
 Symbol Column • 125
 Synchronizing The Real Time Clock • 440

T

Tag Column • 491
 Target Configuration Tab • 37
 TARGET Menu • 28, 58
 Task Period • 498
 TCP Port • 240
 TCP/IP Button • 202
 TCP/IP Client Limit • 239
 TCP/IP Max Sockets • 40
 TCP/IP Max TCP Retry Time • 40
 TCP/IP Port Parameters Reference • 208
 Technical Considerations • 608
 Telnet Max • 395
 Telnet Port • 395
 Telnet Timeout • 395
 Template File • 120
 Text • 127, 413
 Text Attributes • 396, 399, 402, 404, 408, 409, 411
 Text Message Interface (TMI) • 63, 211, 483
 Text Message Interface (TMI) Command Reference • 488, 491
 Text Message Interface (TMI) Reference • 490
 Text Message Log Enable • 213
 Textual User Interface (TUI) • 63, 385
The ScadaBuilder Hierarchy • 73, 77, 91, 103, 129, 143, 159, 165, 171, 183, 191, 199, 203, 207, 211, 214, 241, 249, 252, 254, 291, 293, 302, 303, 305, 310, 316, 320, 332, 333, 341, 342, 343, 348, 349, 350, 352, 367, 370, 374, 377
The ScadaBuilder User Interface • 53
 Threshold • 413
 Threshold Constant • 150
 Threshold Register • 150
 Tick Button • 95, 100
 Time • 258
 Time and Date - Format • 409
 Time and Date - Read Only • 409
 Time Button • 87, 95, 100, 175, 373
 Time Format • 83

Time Zone Map/Constant • 212, 495
 Timer Mode • 149
 Timer Period • 149
 Timings Tab • 224
 Title • 406
 To • 372
 To Buffer • 372
 Toolbar Background Color • 461
 TOOLS Menu • 56, 57, 65, 78, 286
 Total File Bytes • 236
 Total Xfer Bytes • 236
 Totalize The Time A Boolean Is True -- Runtime() • 584
 Trace Line Width • 461
 Traces Button • 468, 471
 Traces Shadow Depth • 461
 Track And Hold An Analog Control Value -- trackhld() • 585
 Trail Delay • 201
 Transfer Fail Map • 280, 379
 Transfer Success Map • 280, 379
 Transmit And Receive Buffer Size • 201
 Transmit Connect Delay • 205
 Trend Graphs • 447
 Trend Parameters • 458
 Trending • 447, 455
 Trigger • 94, 157, 281, 373, 380
 Trigger - Options Tab • 151
 Trigger Add Button • 94
 Trigger at Startup • 94
 Trigger Enable Control Map • 151
 Trigger Init Delay • 36
 Trigger Scan Rate • 35
 Triggers Reference • 148
 TUI - Alarm Settings • 401
 TUI - Bar Graph Settings • 401
 TUI - Buffer Field Settings • 403
 TUI - Button Settings • 404
 TUI - Log Settings • 405
 TUI - Page Link Settings • 406
 TUI - Register Field Settings • 407
 TUI - Text Settings • 408
 TUI - Time and Date Settings • 409
 TUI - Value List Settings • 410
 TUI Checkbox • 124
 TUI Designer - Label List Editor • 411
 TUI Designer - Value List Editor • 412
TUI Local HMI (Pinnacle Controllers) • 415
 TUI Text • 408
 Type • 119, 127, 148
Typographical Conventions • 17

U

- Unack Attr • 401
- Unack Mode • 167
- Unack Text • 401
- Unacknowledged Text • 101
- Understanding Bricknet Routing • 355
- Understanding Modbus Routing (Store and Forward) • 269, 270
- Understanding SDX Routing • 296
- Units • 186
- Units Column • 491
- Unload (Current Modem Driver) • 229
- Unmap Button • 135
- Unpack 16 Booleans From An Integer Register -
- Unpack16() • 573
- Unscale Button • 136
- Use Active Authentication to Establish Data
Link • 237
- Use Address Segment From Switch • 39
- Use Authentication • 208
- Use BCC Instead of CRC • 239
- Use CGI Checkbox • 46
- Use Login Authentication • 237
- Use The Integer Register Dictionary As A FIFO
Log Or Trend -- Logana() • 598
- Use The Real Register Dictionary As A FIFO
Log Or Trend -- Logreal() • 596
- Use UDP Instead of TCP • 240
- User ID • 43, 45
- User ID / Password • 209
- User Login Buffer • 188
- User Login Map • 490
- User Portal (Web Interface) • 46, 125, 423, 456,
467
- User Portal Permissions • 428
- Username • 492
- Users Tab • 492
- Using a Cellular Modem Network Port • 211,
241, 301, 483, 490
- Using a Dialup Network Port • 203, 241, 651
- Using Alarm Dialers • 58, 63, 169, 217, 366
- Using Alarms • 63, 159, 448, 486
- Using Bricknet To Communicate With I/O
Modules • 361
- Using I/O Channels and Mapping Registers • 63,
129
- Using Local Alarms • 63, 165
- Using Local Events • 155**
- Using Log Alarms • 81, 97
- Using Log Events • 81, 91

- Using Log Files -- Data and Alarm Logging • 63,
77
- Using More Complex TMI Grouping • 485
- Using Network Destinations • 216, 241
- Using Network Message Links • 198, 217, 223,
272
- Using Network Ports • 63, 199, 241
- Using Network Sessions • 63, 214, 219, 241
- Using Registers in Your Program • 114
- Using Remote Scaling • 381
- Using TCP/IP Over Serial and Dialup • 207
- Using the Voice User Interface • 58, 63, 183
- Using Triggers • 143**

V

- Value • 157
- Value Attributes • 397, 399, 403, 408
- Values • 410
- Variable Access Functions • 591
- Voice Mode • 182
- Voice Rate • 182
- Voice Type • 182
- Volume • 182

W

- Watch Dog Timeout • 36
- Web Portal Button • 46
- Width • 402, 403, 405
- Write A Byte Out To I/O Space -- OutByte() •
570
- Write A Byte To The Transmit Buffer --
ComWrBt() • 510
- Write A Message Register (STRING) To A File
-- fm_write() • 533
- Write A Message Register to a Line in a Text
File -- fm_writeCrLf() • 531, 533, 537
- Write A Message To The Transmit Buffer --
ComWrStr() • 512
- Write A Real Register To A File -- fr_write •
534
- Write A Word Out To I/O Space -- OutWord() •
571
- Write An Integer To A File -- fa_write() • 530
- Write Checkbox • 44
- Write Data To a Socket -- SockWrite() • 632
- Write Multiple Bytes To The Transmit Buffer --
ComWrBts() • 510
- Write The Current RTC Seconds Since 00
00 01/01/70 -- RTCSecWr() • 601
- Write The Date And Time To The RTC From
Integers -- Datewr() • 600

Write The Time To The RTC From Integers --
Timewr() • 600

Write To A Boolean Register By Index --
BooWr() • 593

Write To A Message Register By Index --
MsgWr() • 594

Write To A Real Register By Index -- RealWr()
• 592

Write To An Integer Register By Index --
AnaWr() • 591

Z

Zooming (PC Browsers) • 480