AADvance

AADvance Controller

Demo Unit User Manual

ISSUE 1.2

DOCUMENT: 553850

SSB Technology

Rockwell Automation

Notice

Disclaimer

It is not intended that the information in this publication covers every possible detail about the construction, operation, or maintenance of a control system installation. You should refer to your own (or supplied) system safety manual, installation instructions and operator/maintenance manuals.

Revision and Updating Policy

This document is based on information available at the time of its publication; however, the document contents are subject to change from time to time. You should contact Rockwell Automation Technical Support by e-mail — support@icstriplex.com to check if you have the latest version of this publication.

Your delivery document will state the software release level of your demo kit.

Contents

This page intentionally left blank

Chapter 1

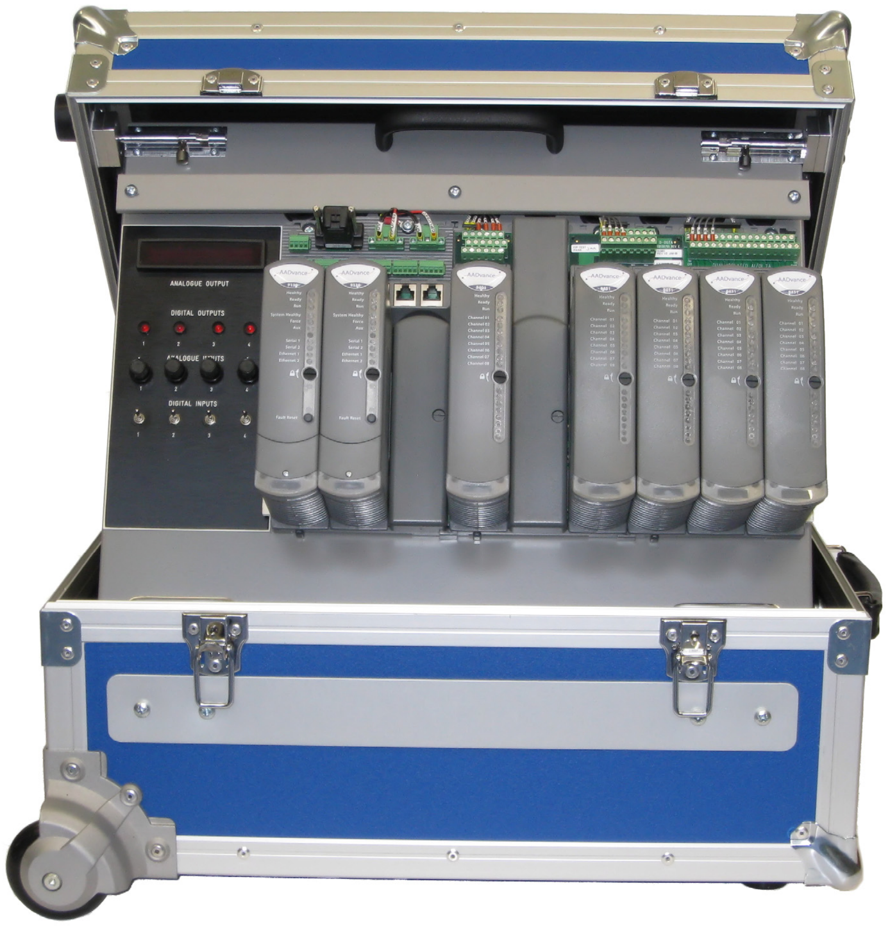
Introduction - Demo Unit

This chapter presents an overview of the demo unit.

In This Chapter

Demo Unit

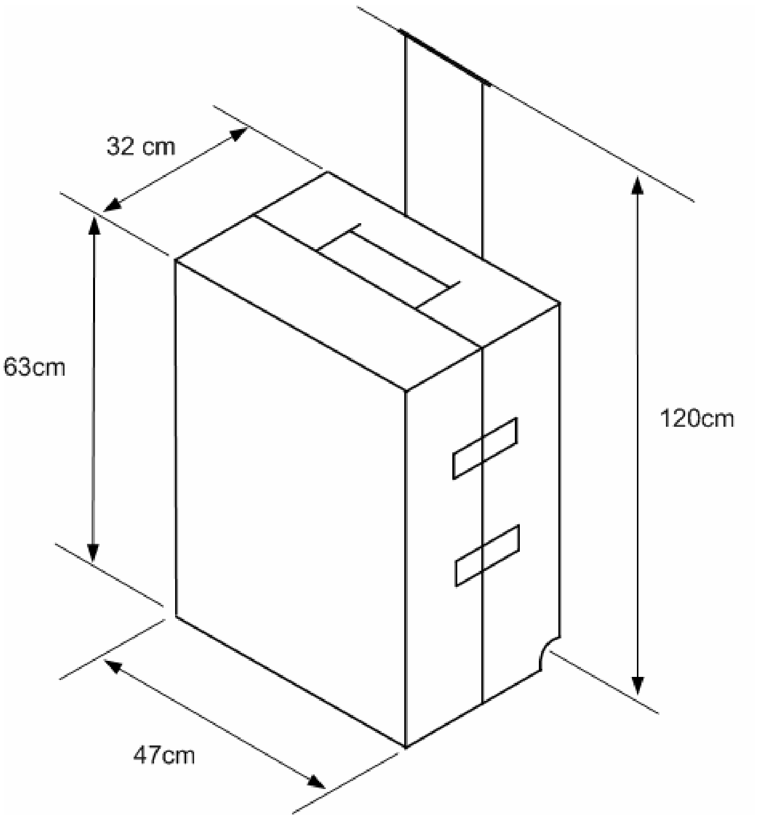
The unit is mounted inside a secure protective box. The box is portable and when it is placed on a flat surface the lid can be lifted and secured in the open position with two bolts. It fully self contained and comes supplied with a set of modules and internal wiring to a display and control panel.



Dimensions and Weight

Dimensions

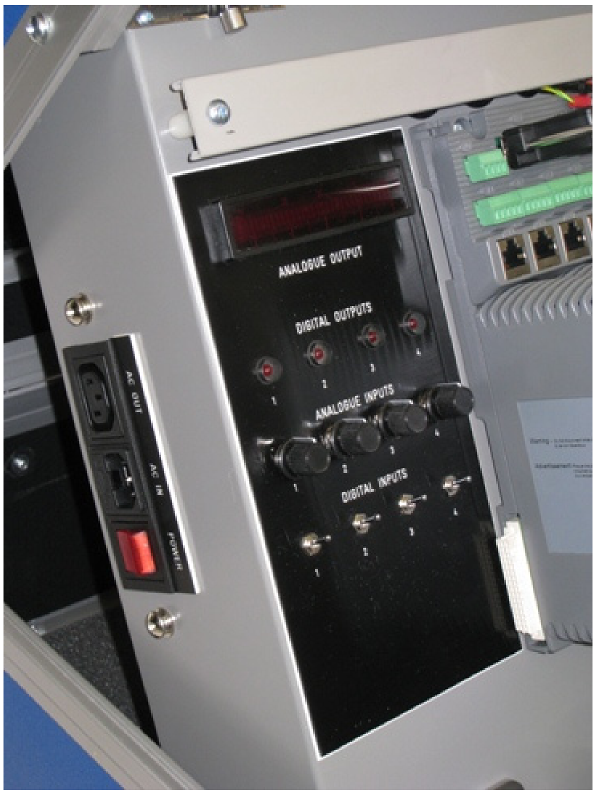
The portable unit has the following dimensions:



Weight

The unit weighs approximately 18 Kgms.

Display Panel and Power Connectors



The display panel provides a visual indication from the analogue and digital output modules; it also has switches to drive digital inputs and potentiometers for analogue inputs signals.The four switches are connected to channels 1 to 4 of the 9401 digital input modul. The four analogue potentiometers are connected to channels 1 to 4 of the dual 9431 analogue input modules. The LED indicators are connected to channels 1 to 4 of the dual digital output modules.

Two power sockets are located on the side of the display panel - one is for the input mains power and the other output socket can provide power for external equipment. The unit can be powered by 240V ac or 110V ac.

Note: The analogue out display is currently not used.

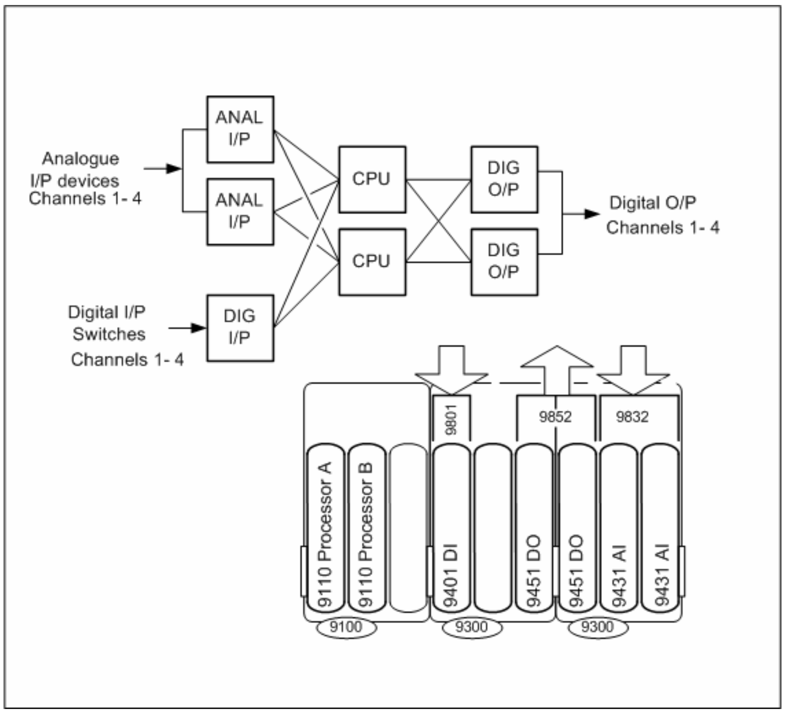
Overview

The Demo Unit is supplied with the following modules and termination assemblies:

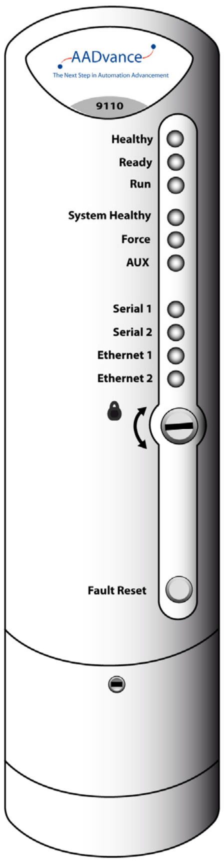
Part No:	Title
T9141	AADvance Demo Unit
T9110	Processor Module
T9110	Processor Module
T9000	Processor Base Unit
T9300	I/O Base unit (3-way)
T9401	Digital Input Module, 24V dc, 8 channel
T9801	Digital Input TA, 16 channel, simplex
T9451	Digital Output Module, 24V dc, 8 channel
T9451	Digital Output Module, 24V dc, 8 channel
T9852	Digital Output TA, 16 channel, Dual
T9431	Analogue Input Module, 8 channel
T9431	Analogue Input Module, 8 channel
T9832	Analogue Input TA, 16 channel, Dual

Hardware Configuration

The controller is configured as follows:



T9110 Processor Module



The T9110 processor module is the central processing unit of an AADvance controller. The processor module carries out the following critical process and safety controller tasks:

- Execution of the AADvance Safety Kernel to solve application logic
- Interfacing with the controller I/O modules, reading and processing input data and writing output data
- Communication with other processor modules, both locally and across the control network
- Initiation of periodic diagnostics for the controller
- Communication with other systems such as HMIs
- Message encapsulation and verification for secure channel communication to other nodes

The processor module is galvanically isolated from external power supplies and data links so that any faults developed in the field cannot cause the module to fail. The module will continue to operate in the event of failure of one of its dual redundant 24V dc power supplies. The module incorporates under- and over-voltage protection for its internal power supplies, which provide a 'power valid' signal to the modules own diagnostics microprocessor.

A processor module has two functionally independent, electrically isolated Ethernet ports. Each port is separately configurable for multiple protocols such as Modbus RTU, Open Modbus/TCP and proprietary AADvance protocols, and its data is available to every processor in the controller.

Two serial communications ports per processor are provided for Modbus RTU slave communications. These ports are also functionally and electrically isolated from each other. They support RS-485 (4– and 2–wire) communications and can be configured to support asynchronous data rates from 1,200 to 115,200 baud.

The processor periodically initiates internal diagnostic tests which, together with a watchdog circuit, monitor the processor internal performance. If the tests detect a serious fault, the processor module will shut down. A controller can use one, two or three processor modules. Using two or three processor modules provides a fault tolerant processor architecture.

If a controller uses two or three processor modules, and one processor module develops a fault, plant maintenance personnel can fit a new processor module while the controller is on-line. The new processor module automatically carries out self-education and synchronizes with the other processors. Fault detection and fail-over in redundant processor configurations is automatic and has no impact on controller operation.

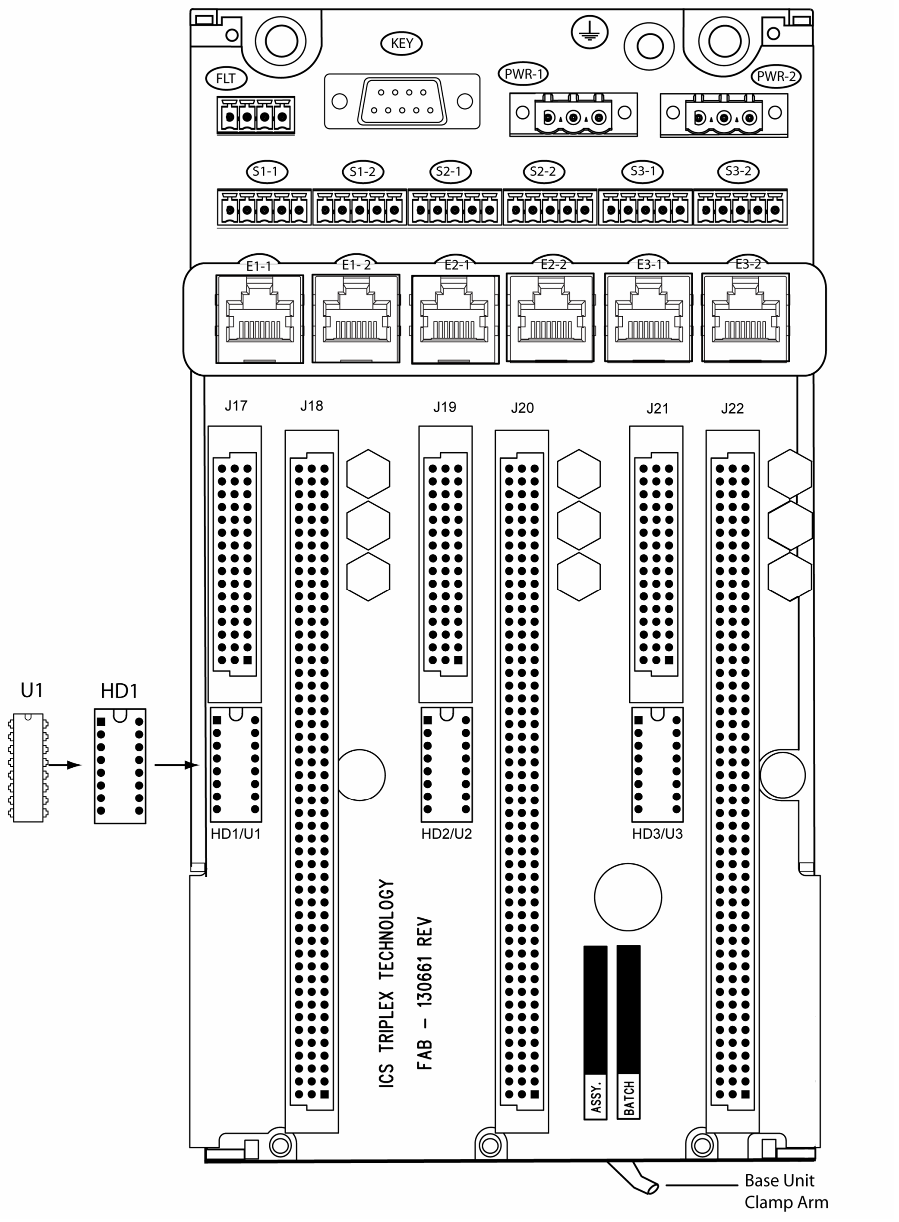
Processor Module Specification

Table 1: Processor Module Specification

Attribute	**Value**
Functional Characteristics	
Degradation	1oo1D, 1oo2D and 2oo3D
Processor clock	400MHz
Memory	
Boot flash	512kB
SRAM	512kB
Bulk flash	64MB
SDRAM	32MB
Sequence of Events	
Event Resolution	Application Scan
Time-stamp Accuracy	5ms
Performance Characteristics	
Safety Integrity Level (SIL)	One module — non-safety applications, SIL1 and/or SIL2 safety applications Two modules — SIL3 applications Three modules — SIL3 fault tolerant and TMR applications
Electrical Characteristics	
Supply Voltage	Redundant + 24V dc nominal; 18V dc to 32V dc range
Power consumption (from 24V dc supply to controller)	6W
Heat dissipation	6W
Maximum Surface Temperature of Module	43°C ± 2°C
Mechanical Specification	
Dimensions (height × width × depth)	166mm × 42mm × 118mm (6-½ in. × 1-5/8 in. × 4-5/8 in.)
Weight	430g (15 oz.)
Casing	Plastic, non-flammable

T9100 Processor Base Unit

Every AADvance controller has one T9100 processor base unit. A processor base unit supports one, two or three modules depending on the architecture chosen for the application.



The processor base unit provides the electrical connections between the T9110 processor modules, and the rest of the controller modules and has the following connections:

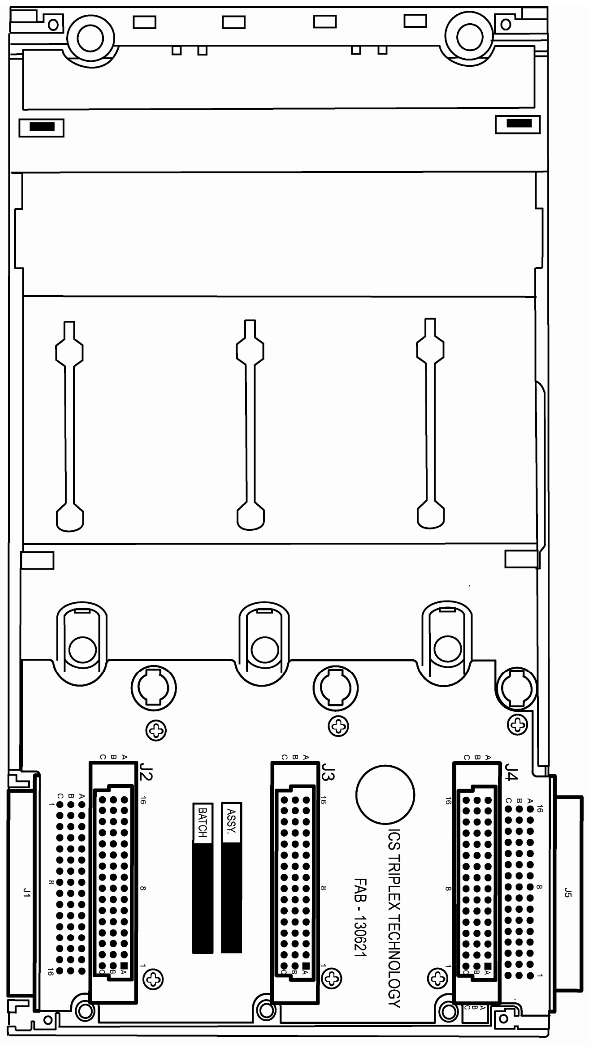
- Command and response bus connections for up to 48 I/O modules
- Inter-processor links
- Two Ethernet 100 BaseT connectors per processor
- Two serial data connections per processor
- Dual +24v System power
- Ground stud
- Program enable key

The processor base unit holds the IP address of each processor module separately in a BUSP. This means that you can remove a defective processor module and install a new one without needing to set up the IP address of the new module.

T9300 I/O Base Unit (3 way)

The AADvance controller has T9300 I/O base units for the I/O modules. An I/O base unit supports up to three I/O modules (of any type), and their associated termination assemblies.

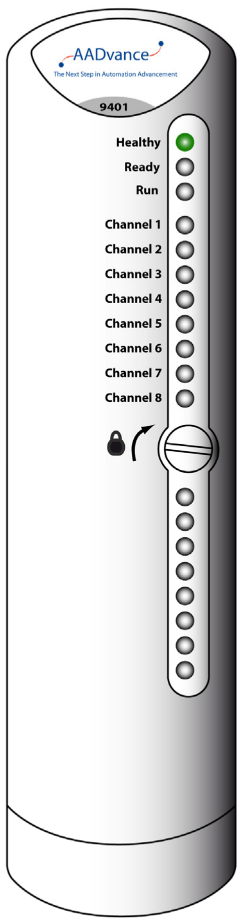
It contains a passive backplane that provides the electrical connections between the I/O modules and the T9100 processor base unit; i.e. the command and response buses and the system power.



The bus and power connections from the processor base unit enter the backplane at the left connector and are routed direct to the module connectors. The backplane provides a connector at the right for the next I/O backplane. The connection to the left of the backplane can connect to a processor base unit or another I/O base unit.

Adjacent base units clip together and are held in position by a plastic retaining clip. Alternatively rows of I/O base units can be connected together using a T9310 expansion cable assembly.

T9401/2 Digital Input Module, 24V dc, 8/16 channel



The T9401/2 digital input module monitors eight (T9401) or sixteen (T9402) isolated digital input channels and notifies the processor module of each field device state and channel condition. Each channel provides both digital state and analogue voltage data to the processor module for field device state, line monitoring and field fault detection.

The input module provides local module and channel status indications through its front panel LEDs, the same indications can be connected to application variables and viewed at the Workbench. Comprehensive diagnostics at both system and module levels generate clear fault indications which help rapid maintenance and repair.

The module incorporates signal and power isolation circuits, which separate each input channel from the rest of the system, protecting the controller from field faults. An independent watchdog arrangement monitors the module operation and provides additional fault containment by a shutdown mechanism should a fault occur.

When a controller uses a digital input module in a dual or TMR configuration, plant maintenance personnel can fit a new input module without interrupting the input signals.

Digital Input Line Monitoring

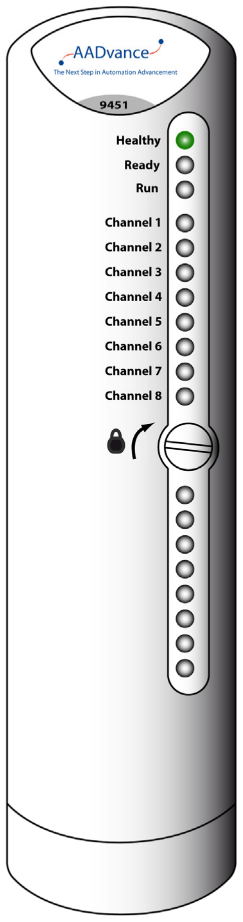
Each digital input module parameters is set up through the AADvance Workbench configuration tools. Switching levels for each digital input channel are configurable at the module and the channel level. Each input has five configurable voltage bands (there are eight distinct switching thresholds to allow hysteresis), each of which can be adjusted through the AADvance Workbench to provide line monitoring, field loop monitoring and additional field device diagnostics.

T9401/2 Digital Input Module Specification

Table 2: T9401/2 Digital Input Module Specification

Attribute	Value
Functional Characteristics	
Input Channels	T9401: 8 T9402: 16
Degradation	1oo1D, 1oo2D,2oo3D
Performance Characteristics	
Safety Integrity Level	IEC 61508 SIL3
Safety Accuracy Limit	1%
Sample Update interval (no filter)	5ms
Self Test Interval	Not Applicable
Sequence of Events Event Resolution Time-stamp Accuracy	 1ms 5ms
Electrical Characteristics	
Supply Voltage	Redundant + 24Vdc nominal; 18V to 32V dc range
Data Input voltage	+24V dc
Input Voltage Measurement Accuracy	± 0.5V
Module Power Consumption	T9402: 1.5W T9402: 2.2W
Module Heat Dissipation	T9401: 1.5W T9402: 2.2W
Channel Power connsumption	0.11W
Channel Isolation (channel to channel and channel to chassis) Maximum withstand	 ± 1.5KV dc
Mechanical Specification	
Dimensions	166mm x 42mm x 118mm (6½ in. × 1 21/32 in. × 4 21/32 in.)
Weight	T9401: 280g (10 oz.) T9402: 340g (12 oz.)
Casing	Plastic, non-flammable

T9451 Digital Output Module, 24V dc, 8 channel



The T9451 digital output module interfaces up to eight final elements and can switch 1A at 32V dc for each device. It features voltage and load current monitoring on each channel, reverse current protection and short and open circuit line monitoring. It is designed to always be able to switch off an output when demanded. No single failure within the module can cause a stuck-on failure.The module supports dual redundant power feeds for field devices without the need for external diodes.

The output module isolates the processor module from the output channel control and data management circuits, thus protecting the processor module from potential faults in the output control circuits and field connections.

The output channel protection activates when the channel load exceeds a safe limit. The reverse voltage protection circuit in each output channel ensures that externally applied voltages do not generate current flow into the module outputs.

The module has self-checking functionality. Short circuit and open circuit line monitoring is provided on all outputs. Internal diagnostics carry out ongoing functionality checks ensuring that the output channel command data is correctly transferred to the output. In addition, the processor module initiates a test sequence on each output channel, checking for 'stuck-on' and 'stuck-off' conditions on the output switch pairs.

Front panel LEDs provide module, channel and field connection status indications. These status indications can be connected to application variables and viewed at the AADvance.

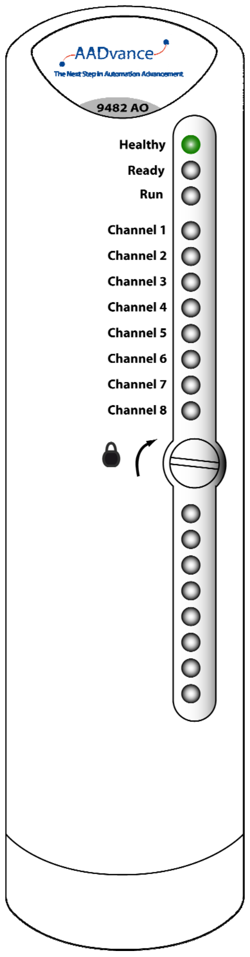
When a controller uses a pair of digital output modules in a dual configuration, the two fail-safe output switches on each channel are combined in a parallel arrangement so that they automatically form a fault-tolerant output configuration.

T9451 Digital Output Module Specification

Table 3: T9451 Digital Output Module Specification

Attribute	**Value**
Functional Characteristics	
Number of output channels	8 per module
Degradation	1oo1D, 1oo2D
Performance Characteristics	
Safety integrity level	IEC 61508 SIL3
Self-test interval	<30 mins (30s per module)
Electrical Characteristics	
Supply Voltage	Redundant +24V dc nominal; 18V dc to 32V dc range
Output characteristics:	
Operating field supply voltage	0V to +50V dc
Maximum voltage without damage	–1V to +60V dc
Nominal output voltage	+ 24V dc
Range	+ 18V to 32V dc
Output current	1A continuous per channel
Voltage Drop at Maximum Current	< 1volt (approximately 0.9V)
Max output current before shutdown	6 A @ 60°C for all channels
Output overload protection	
Surge	10A for up to 50ms
Continuous	1.5A
Power consumption	
Module power (from controller 24V supply)	2W
Channel Field power (from source of field power)	24W (up to 192W per module)
Total maximum power consumption	198W (all 8 channels 'on' at maximum current)
Heat dissipation	6W per module
Mechanical Specification	
Dimensions (height × width × depth)	166mm × 42mm × 118mm (6-½ in. × 1-21/32 in. × 4-21/32 in.)
Weight	340g (12 oz.)
Casing	Plastic, non flammable

T9431/2 Analogue Input Module, 8/16 Channel



The T9431/2 analogue input module monitors eight (T9431) or sixteen (T9432) isolated analogue input channels and notifies the processor module of the field device value and channel condition. Each channel provides digital state and analogue process value data to the processor for process monitoring, line monitoring and field fault detection.

The input module provides local module and channel status indications through its front panel LEDs, the same indications can be connected to application variables and viewed at the Workbench. Comprehensive diagnostics at both system and module levels provide clear indications which help rapid maintenance and repair.

The module incorporates signal and power isolation circuits, which separate each input channel from the rest of the system, protecting the controller from field faults. An independent watchdog arrangement monitors the module operation and provides additional fault containment by a shutdown mechanism should a fault occur.

When a controller uses an analogue input module in a dual or TMR configuration, plant operations personnel can fit a new input module without interrupting the input signals.

Analogue Input Line Monitoring

Each analogue input module is set up through the AADvance Workbench. Monitoring levels for each analogue input channel are configurable at the module and the channel level. The default parameters are

- Fault: 0 to 3.8mA
- Normal: 3.8 to 22.0mA
- Fault: > 22.0mA

Each input has five configurable voltage bands (there are eight distinct switching thresholds to allow hysteresis), each of which can be adjusted to provide line monitoring and field device diagnostics.

T9431/2 Analogue Input Module Specification

Table 4: T9 Analogue Input Module Specification

Attribute	Value
Functional Characteristics	
Number of Field connections	16
Modules Supported:	T9431: 8 channels T9432: 16 channels
Degradation	1oo1D, 1oo2D and 2oo3D
Performance Characteristics	
Safety integrity level	IEC 61508 SIL3
Self test interval	Not Applicable
Safety Accuracy	1%
Electrical Characteristics	
Supply Voltage	Redundant +24V dc nominal, 18V dc to 32V dc
Input Current	
Nominal	4 to 20mA dc
Maximum range	0 to 24mA dc
Resolution	0.0039mA (12 bits over 4 to 20mA range)
Measurement accuracy at 25°C	± 0.05mA
Channel field power (from source of field power)	75mW (based on a 25mA analogue loop terminated by 120Ω)
Module power consumption	T9431: 1.5W T9432: 2.2W
Module Heat dissipation	T9431: 1.5W T9432: 2.2W
Channel Heat dissipation	0.06W
Channel Isolation (channel to channel and channel to chassis)	
Maximum withstanding	± 1.5kV dc
Mechanical Specification	
Dimensions (height × width × depth)	166mm × 42mm × 118mm (6-½ in. × 1-21/32 in. × 4-21/32 in.)
Weight	T9431: 280g (10 oz.) T9432: 340g (12 oz.)
Casing	Plastic, non-flammable

This page intentionally left blank

Chapter 2

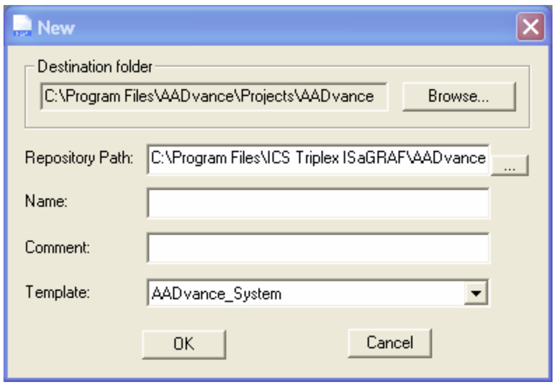
Setting Up the Demo Unit

This chapter will describe the process to set up the Demo Unit ready for configuring the processor and I/O modules.

In This Chapter

Create a New Project

The configuration process starts by creating a new AADvance project. To create a new project do the following:

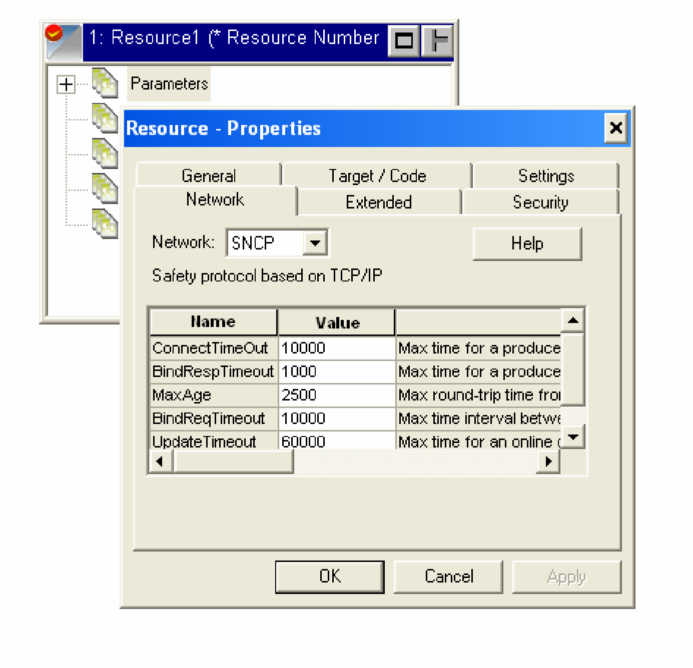

New
Destination folder: C:\Program Files\AADvance\Projects\AADvance
Repository Path: C:\Program Files\ICS Triplex ISaGRAF\AADvance
Name:
Comment:
Template: AADvance_System

1) Start the AADvance AADvance Workbench.
2) Select **File** then **New Project/Library (<ctrl>+N)**.
   - The New dialog box opens.
3) Enter a project name (maximum of 128 characters) and add a comment line.
4) Choose the **AADvance_System** template from the drop down list, click **OK**.
   - The AADvance Workbench creates a project.

Changing the Properties of a Resource

Note: This procedure is only for Release 1.1 versions of the Controller.

If you change any properties of a Resource (see illustration), you have to clean the project/library before you recompile the project. Do the following:



Network: SNCP

Safety protocol based on TCP/IP

Name	Value	
ConnectTimeOut	10000	Max time for a produce
BindRespTimeout	1000	Max time for a produce
MaxAge	2500	Max round-trip time fro
BindReqTimeout	10000	Max time interval betw
UpdateTimeout	60000	Max time for an online

1) Select the **Network** tab on the Resources-dialog.
2) Click on each field in turn and delete the current value (using the delete or backspace key).
3) Enter the following default values (or your own values).
   - **Connect TimeOut** = 10000
   - **BindResp Timeout** = 1000
   - **MaxAge** = 2500
   - **BindingReq Timeout** = 10000
   - **Update Timeout** = 60000
4) On the Resource - Properties dialog, click **OK** to save your changes.
5) On the main menu of the AADvance Workbench, select **Project** → **Clean Project/Library**.
6) You can now choose to recompile your project.

Allocate IP Addresses for Network Communications

The AADvance system uses Internet Protocol (IP) for all communications between the controller and the AADvance Workbench. This includes downloading the application to the controller and real-time monitoring of the system in operation.

For many systems, the administrator of the local area network will allocate the address for the controller. If this is not the case, choose an address from the ranges allocated to private networks:

- 10.0.0.0 to 10.255.255.255 (10/8 prefix)
- 172.16.0.0 to 172.31.255.255 (172.16/12 prefix)
- 192.168.0.0 to 192.168.255.255 (192.168/16 prefix)

Each controller on a particular local area network must have a unique IP address.

Note: You must ensure that the two Ethernet ports on each T9110 processor module are on different subnets.

Example

As an example you can use subnet masks to ensure that the two ports on a processor module are on different subnets:

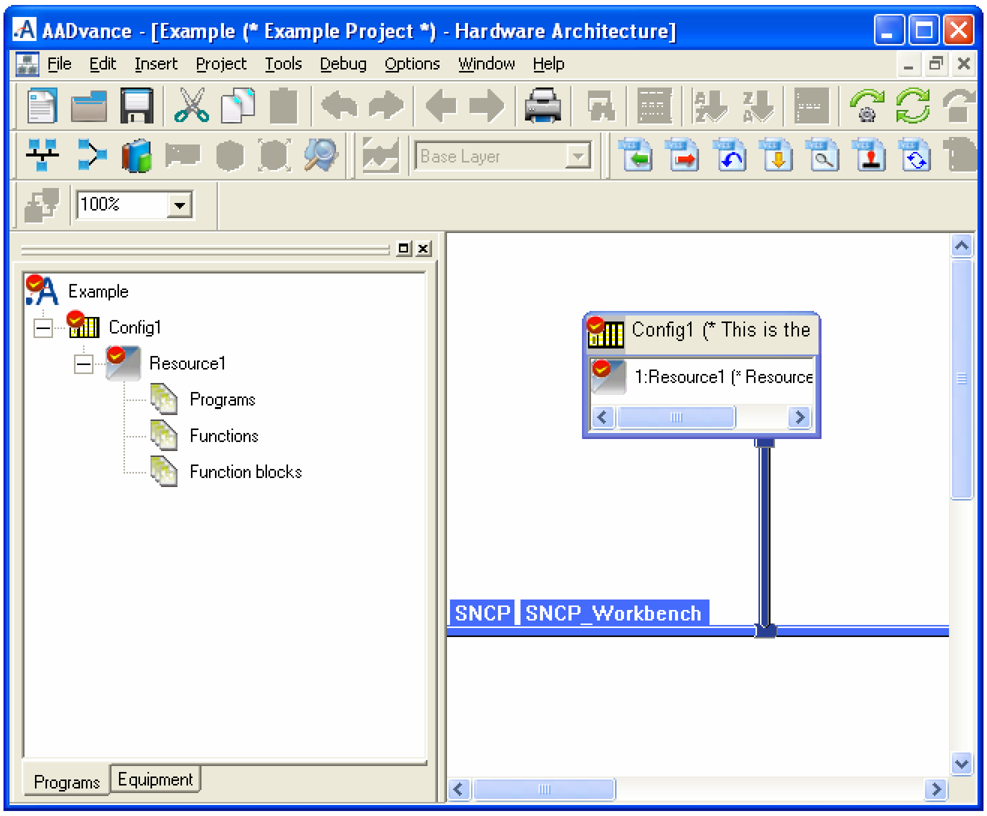
Ethernet port E1-1 Address: 10.10.1.1
Subnet Mask: 255.255.255.0
Ethernet port E1-2 Address: 10.10.2.1
Subnet Mask: 255.255.255.0

The subnet mask defines the first three digits of the IP address, in this case 10.10.1 and 10.10.2.

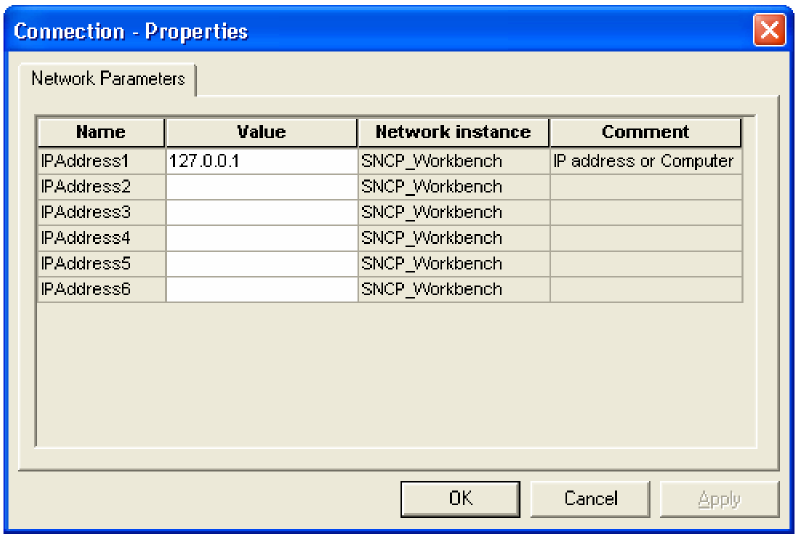
Configure the IP Address of the Target Controller

To connect the AADvance Workbench project to the target controller you have to tell the project the IP addresses allocated to the controller. Do the following:

1) Select the Hardware Architecture view then double-click on the vertical connection between the SNCP network line and the configuration.



2) The **Connection - Properties** dialog box opens.



Name	Value	Network instance	Comment
IPAddress1	127.0.0.1	SNCP_Workbench	IP address or Computer
IPAddress2		SNCP_Workbench	
IPAddress3		SNCP_Workbench	
IPAddress4		SNCP_Workbench	
IPAddress5		SNCP_Workbench	
IPAddress6		SNCP_Workbench	

3) Enter the IP Addresses in the **Value** field for each of the required Ethernet network. Press Enter after typing each IP address.

Note: The value shown above is a default value. Enter a value that you require.

4) Click **OK.**

You have now configured the IP addresses of the configuration to match the controller.

Chapter 3

Downloading the Application to the Controller

This chapter describes the procedures for connecting the AADvance Workbench to the controller so that the application can be downloaded.

In This Chapter

Setting Up the Controller for AADvance Workbench Communications

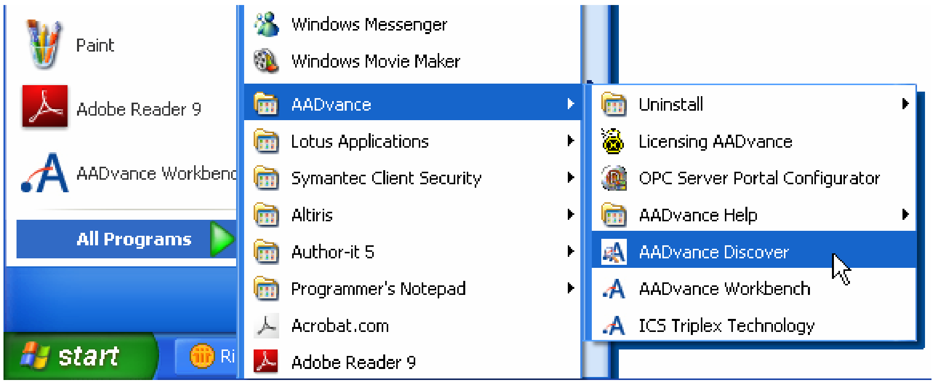
The AADvance controller stores a resource number and IP address information. These details have to match those defined in the AADvance Workbench for the application. After you have configured these details the AADvance Workbench can communicate with the controller. You use the **AADvDiscover** utility to set up the controller for AADvance Workbench communications.

Controller Discovery and Configuration

The **AADvDiscover** Utilility uses a discovery and configuration protocol (proprietary to Rockwell Automation) to set the controller IP address within the AADvance Workbench and to scan the broadcast domain for other AADvance controllers. The utility locates each controller by its unique **MAC Address**. Having located a particular controller to be configured, the utility lets you configure the resource number and **IP Address** to be stored in the controller; after you have done this, the AADvance Workbench can communicate with the other controller.

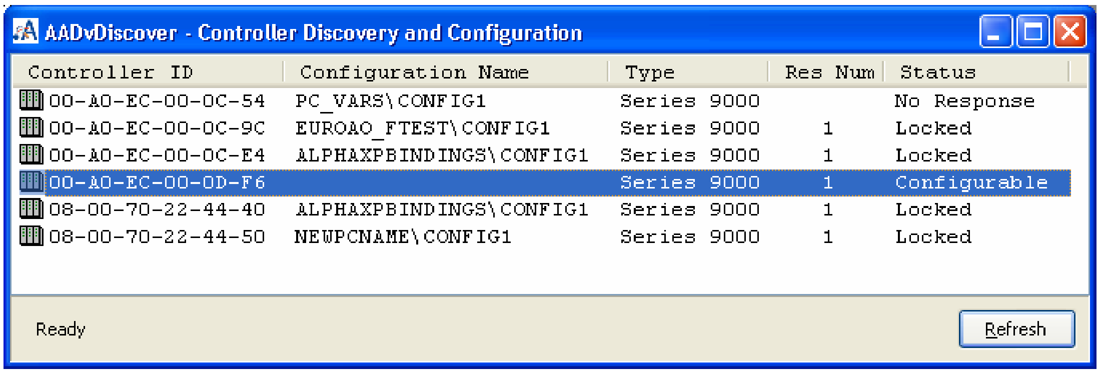
About the AADvDiscover Utility

The AADvDiscover utility is installed when you install the AADvance Workbench, and appears on the Start menu of the computer. Click on **AADvance Discover** to start the AADvDiscover utility.



The **AADvDiscover** utility displays a list of the AADvance controllers on the broadcast network, and reports a status for each one:

- Configurable
- Locked
- No response



Controller ID	Configuration Name	Type	Res Num	Status
00-A0-EC-00-0C-54	PC_VARS\CONFIG1	Series 9000		No Response
00-A0-EC-00-0C-9C	EUROAO_FTEST\CONFIG1	Series 9000	1	Locked
00-A0-EC-00-0C-E4	ALPHAXPBINDINGS\CONFIG1	Series 9000	1	Locked
00-A0-EC-00-0D-F6		Series 9000	1	Configurable
08-00-70-22-44-40	ALPHAXPBINDINGS\CONFIG1	Series 9000	1	Locked
08-00-70-22-44-50	NEWPCNAME\CONFIG1	Series 9000	1	Locked

Double-clicking on an entry in the list lets you inspect the resource and IP address settings for a controller. There is also a **Refresh** button, which makes a scan of the network and creates a new list.

A controller is configurable when the program enable key is present (this plugs into the **KEY** connector on the processor base unit) and either no application is loaded or an application is loaded but not running. The status will be locked if the controller reports that one or more of these criteria has not been met.

If the **AADvDiscover** utility reports a status of 'no response' for a controller, either the controller has been turned off or the communications between the computer running the utility and the controller have failed. Check the power to the controller and check the connection, and click the Refresh button.

The **AADvDiscover** utilty also reports a status of 'in progress' and '**Pending restart**'. '**In progress**' appears while the controller accepts new settings. '**Pending restart**' means the controller is waiting for manual intervention from you; cycle the power to the controller.

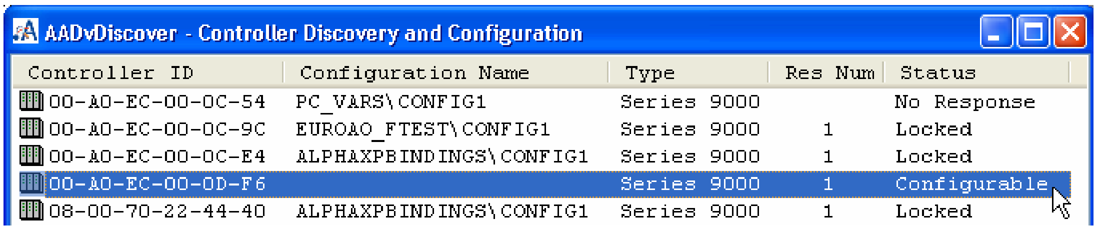
Configure the Controller Resource Number

When you build a new AADvance controller, or install a new 9100 processor base unit, you have to configure the resource number stored in the controller. This is a kind of device address, and it must also be configured in the application.

The procedure to configure the resource number uses the **AADvDiscover** utility. To set the resource number do the following:

1) Make a note of the controller's **MAC address** (**Controller ID**); this is shown on a label on the processor base unit. Install at least one 9110 processor module into the processor base unit.
2) Make sure the program enable key is inserted in the **KEY** connector on the processor base unit.
3) Start the AADvDiscover tool from the Start menu:

- **Start** → **All Programs** → **AADvance** → **AADvance Discover**.
- The **AADvDiscover** utility scans the network for controllers, and creates a list.

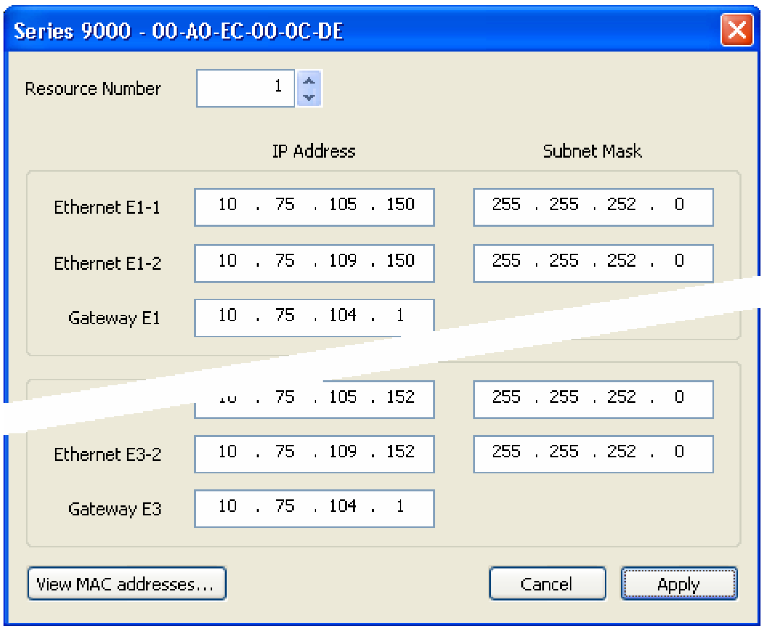


Controller ID	Configuration Name	Type	Res Num	Status
00-A0-EC-00-0C-54	PC_VARS\CONFIG1	Series 9000		No Response
00-A0-EC-00-0C-9C	EUROAO_FTEST\CONFIG1	Series 9000	1	Locked
00-A0-EC-00-0C-E4	ALPHAXPBINDINGS\CONFIG1	Series 9000	1	Locked
00-A0-EC-00-0D-F6		Series 9000	1	Configurable
08-00-70-22-44-40	ALPHAXPBINDINGS\CONFIG1	Series 9000	1	Locked

4) Locate the controller in the list and make sure that the status of the controller is **Configurable**.

5) Double-click on the **MAC address** in the **Controller ID** field.

- The resource and **IP Address** dialog box opens.



Resource Number: 1

	IP Address	Subnet Mask
Ethernet E1-1	10 . 75 . 105 . 150	255 . 255 . 252 . 0
Ethernet E1-2	10 . 75 . 109 . 150	255 . 255 . 252 . 0
Gateway E1	10 . 75 . 104 . 1	
	10 . 75 . 105 . 152	255 . 255 . 252 . 0
Ethernet E3-2	10 . 75 . 109 . 152	255 . 255 . 252 . 0
Gateway E3	10 . 75 . 104 . 1	

6) Enter the resource value into the **Resource Number** field, click **Apply**.

- Returning to the main window of the utility, the controller status will show **Pending Restart**.

7) To complete the update, cycle the power to the controller.

8) Refresh the screen to confirm that the new resource number is displayed in the resource field and the controller status is configurable.

The **Resource Number** must also be configured in the application, in the **Resource Properties**.

Configure the IP Address in the Controller

When you build a new AADvance controller, or install a new 9100 processor base unit, you have to configure the **IP Address** stored in the controller.

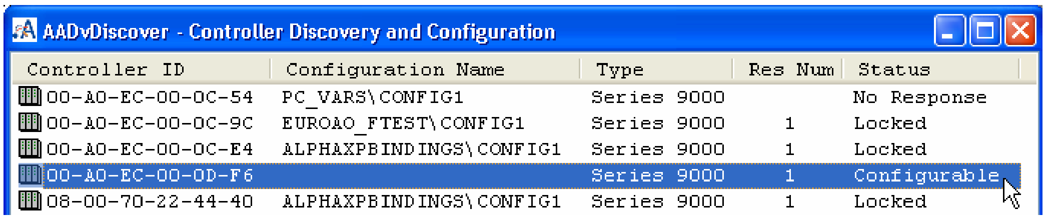
The procedure to configure the **IP Address** uses the **AADvDiscover** utility. Changes take effect immediately and you do not have to restart the controller. To set the **IP Address** do the following:

1) Make a note of the controller's MAC address (**Controller ID**); this is shown on a label on the processor base unit. Install at least one 9110 processor module into the processor base unit.

2) Make sure the program enable key is inserted in the **KEY** connector on the processor base unit.

3) Start the **AADvDiscover** tool from the Start menu:

   - **Start** → **All Programs** → **AADvance** → **AADvance Discover**.
   - The **AADvDiscover** utility scans the network for controllers, and creates a list.



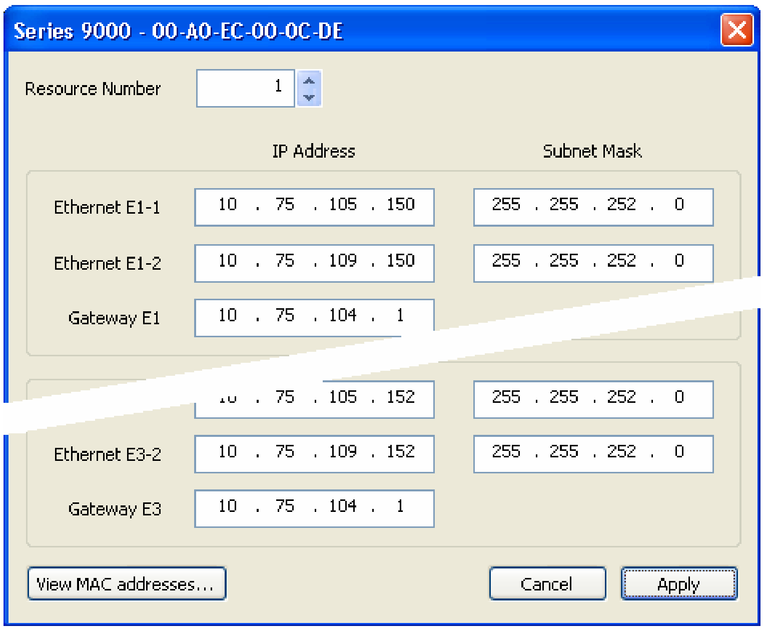
AADvDiscover - Controller Discovery and Configuration

Controller ID	Configuration Name	Type	Res Num	Status
00-A0-EC-00-0C-54	PC_VARS\CONFIG1	Series 9000		No Response
00-A0-EC-00-0C-9C	EUROAO_FTEST\CONFIG1	Series 9000	1	Locked
00-A0-EC-00-0C-E4	ALPHAXPBINDINGS\CONFIG1	Series 9000	1	Locked
00-A0-EC-00-0D-F6		Series 9000	1	Configurable
08-00-70-22-44-40	ALPHAXPBINDINGS\CONFIG1	Series 9000	1	Locked

4) Locate the controller in the list and make sure that the status of the controller is **Configurable**.

5) Double-click on the **MAC address** in the **Controller ID** field.

   - The resource and IP address dialog box opens.



Series 9000 - 00-A0-EC-00-0C-DE

Resource Number: 1

	IP Address	Subnet Mask
Ethernet E1-1	10 . 75 . 105 . 150	255 . 255 . 252 . 0
Ethernet E1-2	10 . 75 . 109 . 150	255 . 255 . 252 . 0
Gateway E1	10 . 75 . 104 . 1	
	75 . 105 . 152	255 . 255 . 252 . 0
Ethernet E3-2	10 . 75 . 109 . 152	255 . 255 . 252 . 0
Gateway E3	10 . 75 . 104 . 1	

6) Enter the **IP Address** and **Subnet Mask** into the fields for each Ethernet port.

7) Enter the **Gateway** values for each processor module, click **Apply**.

   - Returning to the main window of the utility, the controller status will show **In Progress** and then **Configurable**.
   - The controller uses the new settings.

This page intentionally left blank

Chapter 4

Configuring the Controller Processor Modules

This chapter describes the process to configure the processor modules:

In This Chapter

About The Configuration Process

The configuration process for the AADvance AADvance Workbench enables you to configure the controller architecture for your hardware configuration and to connect application variables to I/O points and module status parameters.

The process begins by creating a project and allocating the IP addresses for its communications to the AADvance controllers. You can then configure the network communications parameters for the project.

You then define the hardware architecture. This assigns the I/O modules to empty slot numbers on the processor buses. There are two IO Busses each can be assigned up to 24 I/O modules.

Note: If you change the physical arrangement of the hardware after you have configured a controller using the AADvance Workbench, you must change the AADvance Workbench configuration to match the changed hardware arrangement. However this can only be done when the system is Off-Line and cannot be done to a live system.

You should now define your module status and the I/O channel variables and their properties in the Dictionary. The AADvance Workbench provides you with a wide range of variables types to choose from including a set of structured variables. Set up enough variables to cover all the I/O points and module status variables for your controller architecture. If necessary, you can add new variables at any time during configuration of a system and the AADvance Workbench or after reconfiguration.

You should now allocate tag names to the variables you want to use. If you chose structured variables for I/O channels, the AADvance Workbench automatically generates a set of additional variable elements with the same tag name for each each element type.

In the next stage of the process you define the T9110 processor module functionality and set up connections to a group of processor module status parameters. Here you will enter values for functions such as the serial port settings, process safety time, and SNTP and Modbus services.

The AADvance Workbench provides pre-defined I/O module status parameters for each module to which you assign application variables.

Finally you connect (wire) each I/O channel to structured variables. These structured variables report input the channel status and define output data values.

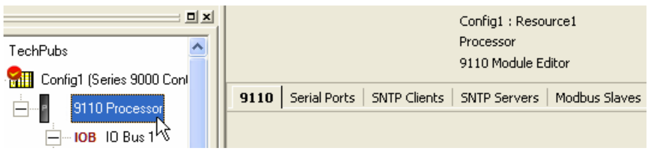
You define hardware redundancy in the AADvance Workbench when you define the hardware configuration. During the allocation of I/O modules to empty slots, you are presented with the option to add two or three modules. When you choose the two or three option the AADvance Workbench automatically allocates the modules to a group of adjacent slots. The AADvance Workbench then only allows you to configure one set of I/O channels to the group.

Note: You do not need to define redundancy for the processors. The AADvance Workbench automatically connects to all three processors after their IP addresses have been set up in the AADvance Workbench.

About the 9110 Module Editor

The **9110 Module Editor** configures all the main processor functionality and enables you to wire status and control variables.

1) Selecting the **Equipment** tab
2) Select **9110 Processor** to open the editor.



The editor provides a set of tabbed pages where you can configure the different items. Each page provides an editor for a set of related configuration items. When you finish setting up the items on a page, click **Apply** to save your changes before you move to another page.

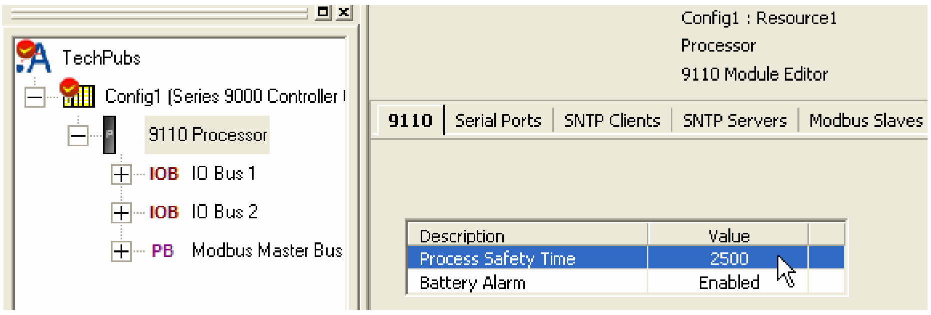
Configure the Top-level Process Safety Time (PST)

The **PST** setting defines the maximum time that the processor will allow the outputs to remain in the ON state in the event of certain internal diagnostic faults or systematic application faults. If **PST** expires the system will go to its safe state.

You have to specify the **PST** for the whole controller. This is a top level setting, which you make once for all the T9110 processor modules.

Note: Groups of I/O modules can inherit this setting, or use individual PST settings instead.

To set the top-level process safety time do the following:



Description	Value
Process Safety Time	2500
Battery Alarm	Enabled

1) Select the **9110 Processor** in the Equipment tree view.
   - The **9110 Module Editor** opens.
2) Select the **9110** tab.
3) Enter the time into the **Process Safety Time** field. Choose from the following range of values:
   - Minimum: 20ms
   - Maximum: 65,535ms (65 seconds)

   Note: If you specify a process safety time below the application execution time, the application will not run. The default is 2,500ms.
4) Click **Apply**.

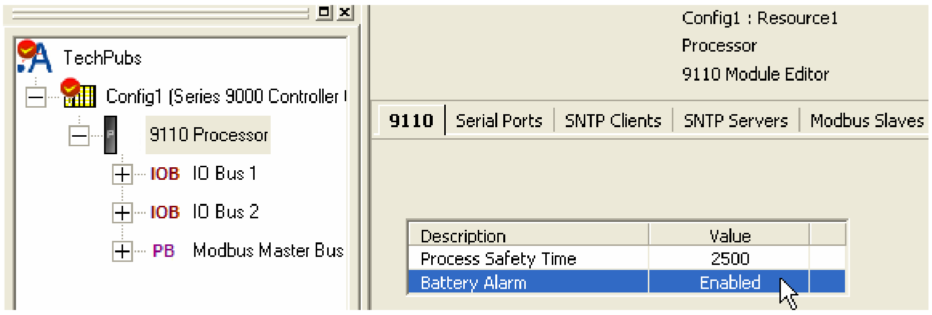
Set PST to its Default Value

You can reset the PST to its default value, do the following:

1) Select the **9110 Processor** in the Equipment tree view.
   - The **9110 Module Editor** opens.
2) Select the **9110** tab.
3) Clear the entry in the **Process Safety Time** field.
4) Press return.
   - The **Process Safety Time** will change to its default value.

Configure the Processor Battery Alarm

The **9110 Module Editor** includes a configuration setting for the battery alarm.



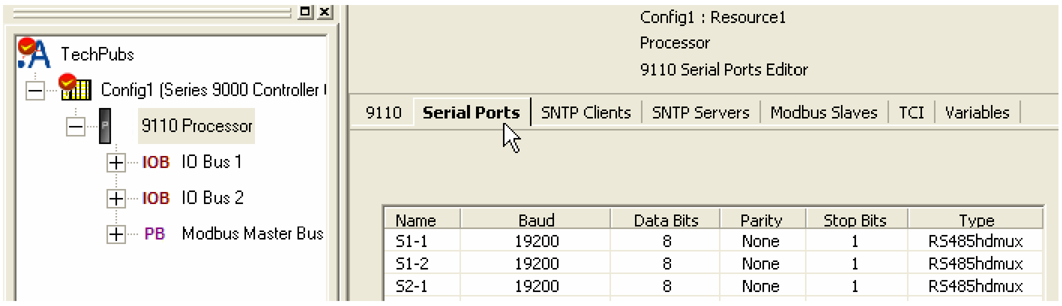
Description	Value
Process Safety Time	2500
Battery Alarm	Enabled

The battery alarm setting has no effect in the current release. Leave this item set to **Enabled**.

Configure the Serial Ports

The AADvance controller provides up to six serial communication ports, two for each T9110 processor module present. .

The serial port settings define the protocol ('type') and the data characteristics of each of the serial ports. To configure the serial ports do the following:



Name	Baud	Data Bits	Parity	Stop Bits	Type
S1-1	19200	8	None	1	RS485hdmux
S1-2	19200	8	None	1	RS485hdmux
S2-1	19200	8	None	1	RS485hdmux

1) Select the **Serial Ports** tab.

   - The **Serial Ports Editor** dialog box opens.

2) Select the communication parameters from the drop down lists, click **Apply**.

3) To restore the default values, click **Default** then **Apply**.

Serial Port Protocols

The serial ports support the protocols listed in the table.

Table 5: Serial Port Protocols

Type	Description
RS485fd	Full-duplex, 4-wire connection with separate buses for transmit and receive
RS485fdmux	Full-duplex, 4-wire connection with separate buses for transmit and receive and tri-state outputs on the transmit connections
RS485hdmux	Half duplex, 2-wire connection

Serial Port Parameters

Each serial port on the AADvance controller supports the set of control parameters as detailed in the table.

Table 6: Controller Serial Port Parameters

Description	Value(s)	Default	Remarks
Baud	2,400, 4,800, 9,600, 19,200, 38,400, 57,600, 76,800 or 115,200	19,200	
Data Bits	5 to 8	8	
Parity	None, Odd or Even	None	
Stop Bits	1 or 2	1	
Type	RS485fd RS485fdmux RS485hdmux	RS485hdmux	'fd' means 'full duplex' 'hd' means 'half duplex'

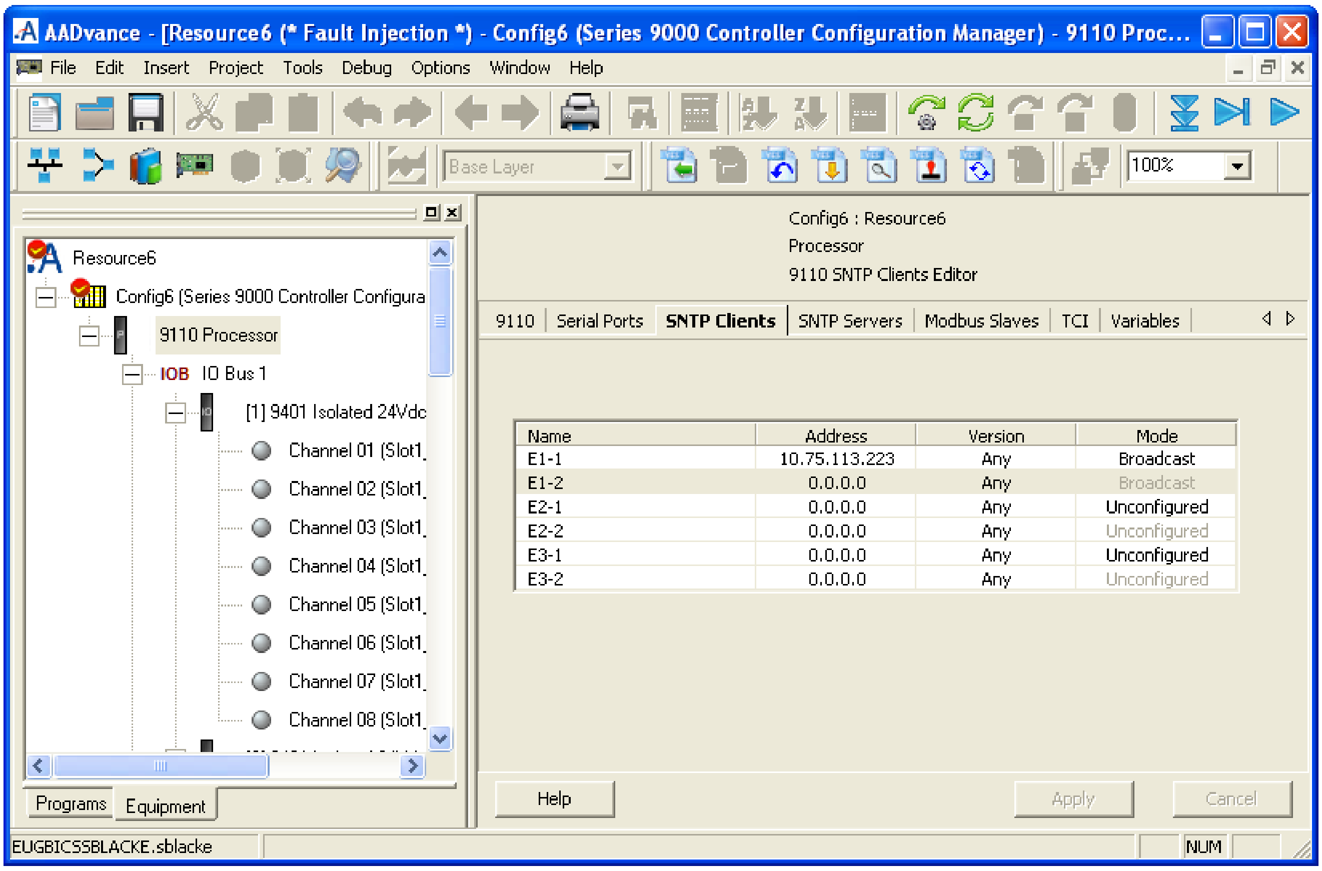
Note: Most systems use two bits after each data byte. The two bits are either a parity bit (odd or even) and one stop bit, or no parity and two stop bits.

Configure the Controller as an SNTP Client

The AADvance controller supports the **Simple Network Time Protocol** (SNTP) service that can circulate an accurate time around the network. As an SNTP client the controller will accept the current time from external **Network Time Protocol** (NTP) and SNTP network time servers.

The SNTP clients settings tell the controller the IP address of the external server; the version of SNTP offered by the server; and the operating mode for the time synchronization signal that the processors will use for their real time clock.

To configure the SNTP clients service do the following:



Name	Address	Version	Mode
E1-1	10.75.113.223	Any	Broadcast
E1-2	0.0.0.0	Any	Broadcast
E2-1	0.0.0.0	Any	Unconfigured
E2-2	0.0.0.0	Any	Unconfigured
E3-1	0.0.0.0	Any	Unconfigured
E3-2	0.0.0.0	Any	Unconfigured

1) Select the **SNTP Clients** tab.

   - The **SNTP Clients Editor** dialog box opens.

2) Set the **E1-1** and **E1-2 Address** fields to the IP addresses of the network time server.

   Note: The first address represents that of the primary server and the second one the secondary server for each processor module. At start up the SNTP client will choose the primary server of the "lowest" slice; if no primary signal is valid the SNTP client looks for an active secondary server signal.

- For non-fault tolerant operation, define one SNTP server for only one processor. The other processors will automatically synchronize to it and will inherit the time.
- For fault-tolerant SNTP client operation, define more than one server address.

3) Select the server version**.**
   - Choose SNTPv1, SNTPv2, SNTPv3, SNTPv4 or Unknown.
   - If you do not know the version of NTP/SNTP that the server offers, choose **Unknown**. This will disable some validation of the incoming signal.
4) Set the **Mode** to **Unicast** or **Broadcast** as required.
   - In **Broadcast** mode the SNTP client will passively wait for regular broadcasts from the server. This reduces network traffic and hence the load on the servers.
   - In **Unicast** mode the SNTP client will actively poll as many servers as are configured every few seconds and use their responses. The polling rate (19s) is based on the drift rate of the real-time clock and cannot be configured.
5) Click **Apply**.

Configure the Controller as an SNTP Server

The AADvance controller can fulfill the role of one or more SNTP servers (one for each processor) to provide a network time signal throughout the network.

To enable serving time on an interface it is necessary to specify the direct broadcast address for that interface. This works for broadcast or unicast modes. This method of configuring is derived from the NTP configuration command language.

For an interface, the directed broadcast address

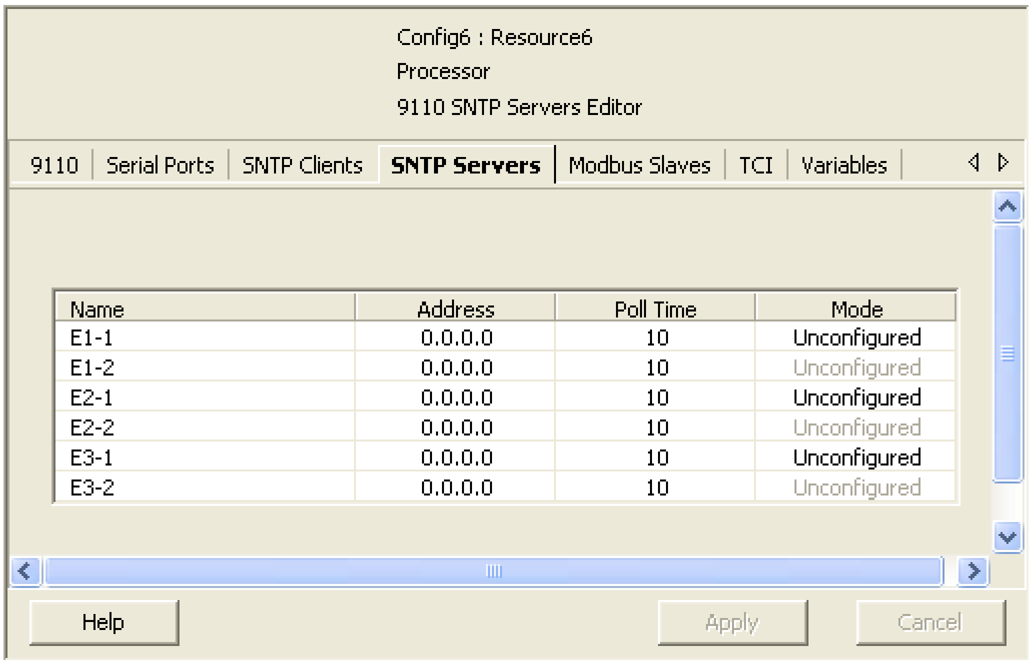
= ((IP address for interface) **bitwise-and** (subnet-mask)) bitwise-or (**bitwise-not** subnet-mask)

For example, if the IP address for an interface is 10.10.1.240 and its subnet-mask is 255.255.255.0 then the directed broadcast address is:

= ((10.10.1.240) **bitwise-and** (255.255.255.0)) **bitwise-or** (**bitwise-not** 255.255.255.0)

= (10.10.1.0) **bitwise-or** (0.0.0.255)

To configure the SNTP servers service do the following:


Config6 : Resource6
Processor
9110 SNTP Servers Editor

Name	Address	Poll Time	Mode
E1-1	0.0.0.0	10	Unconfigured
E1-2	0.0.0.0	10	Unconfigured
E2-1	0.0.0.0	10	Unconfigured
E2-2	0.0.0.0	10	Unconfigured
E3-1	0.0.0.0	10	Unconfigured
E3-2	0.0.0.0	10	Unconfigured

1) Select the **SNTP Servers** tab.
   - The **SNTP Server Editor** dialog box opens.
2) Select the **Unicast** or **Broadcast** mode.
   - If you select **Unicast** mode for a processor the controller will wait to be polled by a client and then respond with a time signal; it will not broadcast any time signals.
   - If you select **Broadcast** mode for a processor the controller will regularly braodcast: it will also respond to unicast polling requests on that interface.

Note: If you set a processor Broadcast IP **Address** to zero (0.0.0.0) it will disable the server on that interface.

3) Set the Broadcast IP **Address** for the network.
4) Repeat steps 2 and 3 for each additional processor module.
5) Click **Apply**.

Using the Controller as a Modbus Slave

The AADvance controller can operate as a **Modbus** slave, supporting up to ten **Modbus** slaves on each 9110 processor module. This gives a capacity of thirty **Modbus** slaves for a controller with three processor modules.

Note: As a **Modbus** slave device, the controller only transmits data upon a request from a **Modbus** master, and does not communicate with other slaves.

Support for Modbus Slave Exceptions

When the AADvance controller operates as a Modbus slave, it can raise these exception codes:

Code 01: Illegal Function

The function code received in the query is not an allowable action for the slave. If a Poll Program Complete command was issued, this code indicates that no program function preceded it.

Code 01 represents a function that the AADvance controller does not recognize or does not support.

Code 02: Illegal Data Address

The data address received in the query is not an allowable address for the slave.

The AADvance controller raises **code 02** when a request specifies an address outside the 16-bit range 0 to 65,535. The exception occurs if the request specifies the address implictly ('give me the 20 registers from address 65,530') or explicitly (give me the register at address 65,536').

Code 03: Illegal Data Value

A value contained in the query data field is not an allowable value for the slave.

The AADvance controller can raise **code 03** only on boolean (coil) writes.

Code 04: Slave Device Failure

An unrecoverable error occurred while the slave was attempting to perform the requested action.

Code 04 represents an internal error within the AADvance controller.

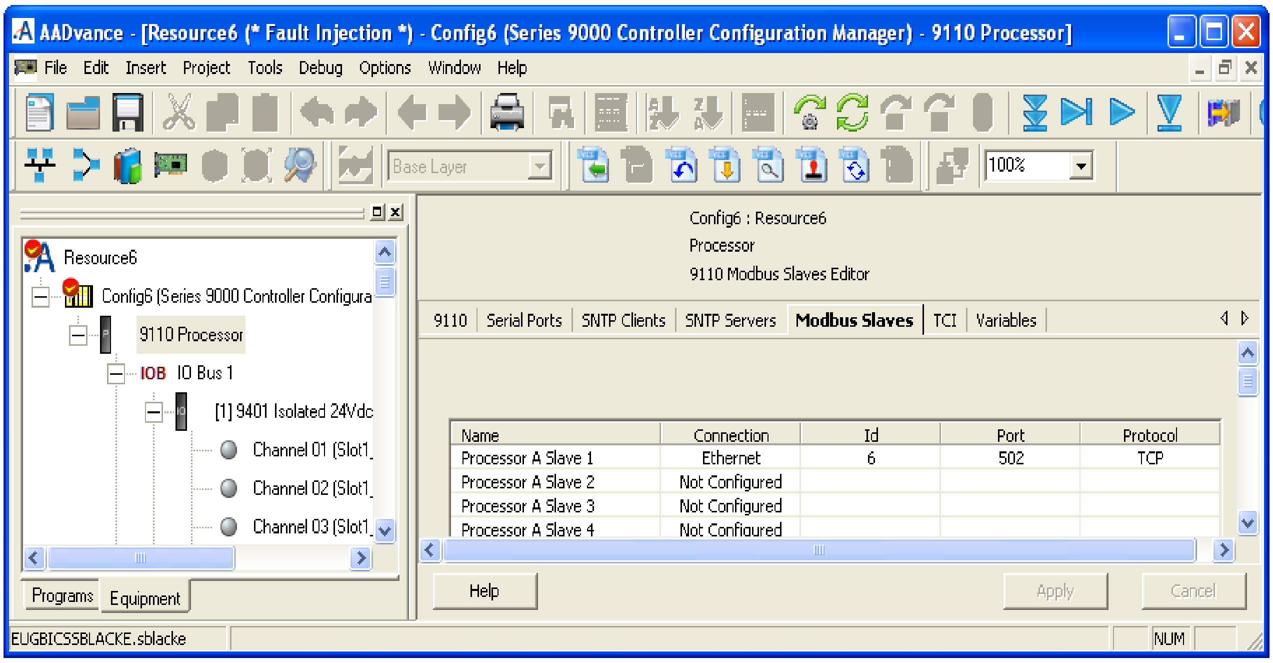
Code 06: Slave Device Busy

The slave is engaged in processing a long-duration program command. The master should retransmit the message later when the slave is free.

The AADvance controller can be 'busy' and thus raise **code 06** while it is waiting for its application to download or to start. The controller can be report itself to be busy for up to 30 seconds; after this period, the controller will cease to respond.

Configure the Controller Modbus Slaves

You have to configure the communication parameters for each **Modbus** slave you implement within the AADvance controller. Do the following:



Config6 : Resource6
Processor
9110 Modbus Slaves Editor

Name	Connection	Id	Port	Protocol
Processor A Slave 1	Ethernet	6	502	TCP
Processor A Slave 2	Not Configured			
Processor A Slave 3	Not Configured			
Processor A Slave 4	Not Configured			

1) Select the **Modbus Slaves** tab.
   - The **9110 Modbus Slaves Editor** dialog box opens.
2) In the **Name** column, locate the processor and slave you wish to configure.
3) Set the Connection field, click **Apply.**
   - The **Id**, **Port** and **Protocol** fields are set to their default values.
4) If you set the Connection to a serial port, the **Id** field represents the Slave ID; set the **Id** field or accept the default value.

   Note: The **Port** field does not apply for a serial connection, and is disabled.
5) If you set the Connection to **Ethernet**, do the following:
   - Set the Protocol field, click **Apply**.

Note: As a **Modbus** slave, the controller supports **Modbus RTU**, using a serial or Ethernet connection; and **Modbus TCP**, using an Ethernet connection. You can configure a combination of connections for the Modbus slaves, subject to a limitation of no more than two **Modbus RTU** slaves using serial communications for each processor.

- The **Id** field represents the Unit ID; set the **Id** field or accept the default value.

- The default setting for the **Port** field suits most systems; occasionally, you will have to adjust it. Make sure the **Port** field matches the port expected by the **Modbus** master.

If the **Port** field is set to 502, **Modbus TCP** protocol is enabled. All other settings of the **Port** field will enable **Modbus RTU** protocol packaged as a serial stream over Ethernet. This can be converted back to a serial connection using a standard terminal server.

Note: The range of values accepted for the **Id** field, and the default value for the **Port** field, vary according to the protocol selected.

1) Click **Apply**.

Modbus Slave Communication Parameters

Each **Modbus** slave has a series of communication parameters as detailed in the tables.

Table 7: Modbus RTU Slave Parameters

Description	Value(s)	Default	Remarks
Connection	Not Configured, Sn-1, Sn-2, Ethernet (†)	Not Configured	
Id	1 to 247 (serial); 1 to 255 (Ethernet)	1 (serial); 255 (Ethernet)	Represents the Slave ID for a serial connection Represents the Unit ID for an Ethernet connection
Port	0 to 65,535	2000	Only used with Ethernet connections

Note: (†) The letter 'n' identifies the processor module:
1 = processor A, 2 = processor B, 3 = processor C.

Table 8: Modbus TCP Slave Parameters

Description	Value(s)	Default	Remarks
Connection	Not Configured, Ethernet	Not Configured	
Id	1 to 255	1	
Port	0 to 65,535	502	

About T9110 Processor Variables

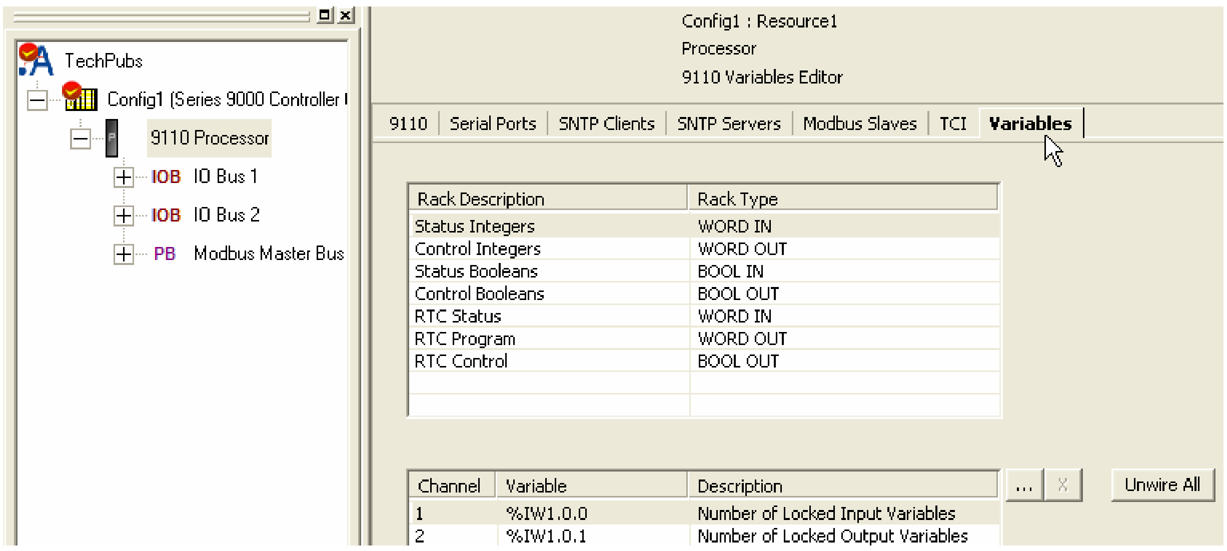
The T9110 processor module provides a number of status and control variables that are available to the application. Status variables retrieve status information; control variables set status information.

The **9110 Variables Editor** presents the variables in seven collections, which it calls 'racks':

- **Status Integers** and **Status Booleans**, which provide information about the controller to the application;
- **Control Integers** and **Control Booleans**, which enable the application to send specific information to the controller;
- **RTC Status variables**, which provide information about the controller real-time clock to the application;
- **RTC Program variables**, which specify parts of the date to be written to the real-time clock;
- **RTC Control variables**, which set and control updates to the real-time clock.

Wire Processor Variables

To wire a **9110 processor** variable do the following:



Rack Description	Rack Type
Status Integers	WORD IN
Control Integers	WORD OUT
Status Booleans	BOOL IN
Control Booleans	BOOL OUT
RTC Status	WORD IN
RTC Program	WORD OUT
RTC Control	BOOL OUT

Channel	Variable	Description
1	%IW1.0.0	Number of Locked Input Variables
2	%IW1.0.1	Number of Locked Output Variables

1) Select the **Variables** tab of the **9110 Processor Editor**.
   - The **9110 Variables Editor** dialog box opens.
2) Select a rack, e.g. **Status Registers**.
   - The editor displays a list of associated channel variables.
3) Select a **Channel**.

4) Click the ... button. The **Select Variable** dialog box opens.

5) From the list select an application variable to wire to the processor variable, click **OK**.

6) Repeat for each subsequent variable to be wired.

Return to the **9110 Processor Editor** dialog box and click **Apply**. The variable will now be wired.

Unwire Processor Variables

To disconnect a 9110 processor variable do the following:

1) Select the **Variables** tab of the **9110 Processor Editor**.
   - The **9110 Variables Editor** will be displayed.
2) Select the relevant rack.
   - The editor displays a list of associated variables.
3) Select the variable to be unwired, click the **X** button.
4) Click **Apply**.
   - The variable will be unwired.

Note: Select the **Unwire All** button and click **Apply** to disconnect all of the wired variables in the rack.

Status Integers

The variables in the rack of status integers provide information about the controller to the application.

Number of Locked Input Variables

Direction: input to application from controller

Type: word

Values:

- 0 to 65,535

Description:
Reports the number of input variables that have been locked by the user. The upper limit of 65,535 represents the capacity of the variable; in practice, the limit is the number of variables in the application.

Number of Locked Output Variables

Direction: input to application from controller

Type: word

Values:

- 0 to 65,535

Description:
Reports the number of output variables that have been locked by the user. The upper limit of 65,535 represents the capacity of the variable; in practice, the limit is the number of variables in the application.

Processor Module A Temperature

Direction: input to application from controller

Type: word

Values:

- 0 to 65,535

Description:
Reports the temperature of the 9110 processor module in the given slot in degrees centigrade. Set to 0 (zero) if no processor module is present.

Processor Module B Temperature

Direction: input to application from controller

Type: word

Values:

- 0 to 65,535

Description:
Reports the temperature of the 9110 processor module in the given slot in degrees centigrade. Set to 0 (zero) if no processor module is present.

Processor Module C Temperature

Direction: input to application from controller

Type: word

Values:

- 0 to 65,535

Description:
Reports the temperature of the 9110 processor module in the given slot in degrees centigrade. Set to 0 (zero) if no processor module is present.

Control Integers

The variables in the rack of control integers enable the application to send specific information to the controller.

AUX LED Colour

Direction: output from application to controller

Type: word

Values:

- 0..3 (0 = off, 1 = red, 2 = green, 3 = amber)
- Default 0

Description:
Sets the state of the LED indicator labelled 'Aux' on every 9110 processor module.

Status Booleans

The variables in the rack of status booleans provide information about the controller to the application.

System Health

Direction: input to application from controller

Type: boolean

Values:

- TRUE = All 9110 processor modules are reporting system healthy and their system healthy LED indicators are green.
- FALSE = One or more processor modules is reporting a system health problem and its system healthy LED indicator is red.

Description:
Reports the state of system health as voted by all processor modules present.

Note: If there are less than three processor modules fitted, the voting process considers a processor module that is absent to be healthy.

System Health Reset (Voted 1oo3)

Direction: input to application from controller

Type: boolean

Values:

- TRUE = The fault reset button on any 9110 processor module has been pressed; valid for one scan only.
- FALSE = No fault reset button is active.
- Default: FALSE

Description:
Reports that the fault reset button on any processor module has been pressed. The system health reset is triggered by pressing the button, but the value does not change to TRUE until the beginning of the next application cycle. The value remains TRUE for the duration of the cycle and then reverts to FALSE, even if the button has been held down throughout.

Dongle Detected (Voted)

Direction: input to application from controller

Type: boolean

Values:

- TRUE = One or more 9110 processor modules is detecting the presence of a program enable key at the KEY connector on the 9100 processor base unit.
- FALSE = No processor module can detect the presence of a program enable key.

Description:
Reports the presence or absence of a program enable key.

Processor Module A On-line

Direction: input to application from controller

Type: boolean

Values:

- TRUE = The 9110 processor module in the given slot is on-line
- FALSE = The processor module is off-line
- Default: TRUE

Description:
Reports that a processor module within a dual or triple modular redundant configuration is present and is communicating through the inter-processor link to one or both of its peers. Reports that a simplex processor module is present.

Processor Module B On-line

Direction: input to application from controller

Type: boolean

Values:

- TRUE = The 9110 processor module in the given slot is on-line
- FALSE = The processor module is off-line
- Default: TRUE

Description:
Reports that a processor module within a dual or triple modular redundant configuration is present and is communicating through the inter-processor link to one or both of its peers. Reports that a simplex processor module is present.

Processor Module C On-line

Direction: input to application from controller

Type: boolean

Values:

- TRUE = The 9110 processor module in the given slot is on-line
- FALSE = The processor module is off-line
- Default: TRUE

Description:
Reports that a processor module within a dual or triple modular redundant configuration is present and is communicating through the inter-processor link to one or both of its peers. Reports that a simplex processor module is present.

Processor Module A Health

Direction: input to application from controller

Type: boolean

Values:

- TRUE = The 9110 processor module in the given slot is healthy and its Healthy LED indicator is green.
- FALSE = The processor module is faulty and its Healthy LED indicator is red.

Description:
Reports the health status of a processor module.

Processor Module B Health

Direction: input to application from controller

Type: boolean

Values:

- TRUE = The 9110 processor module in the given slot is healthy and its Healthy LED indicator is green.
- FALSE = The processor module is faulty and its Healthy LED indicator is red.

Description:
Reports the health status of a processor module.

Processor Module C Health

Direction: input to application from controller

Type: boolean

Values:

- TRUE = The 9110 processor module in the given slot is healthy and its Healthy LED indicator is green.
- FALSE = The processor module is faulty and its Healthy LED indicator is red.

Description:
Reports the health status of a processor module.

Processor Module A 24V1 Power Feed Health

Direction: input to application from controller

Type: boolean

Values:

- TRUE = power feed voltage is within specification (18 to 32V dc).
- FALSE = power feed is outside specification.

Description:
Reports the health of power feed 1 (nominal 24V dc) to the 9110 processor module in the given slot.

Processor Module B 24V1 Power Feed Health

Direction: input to application from controller

Type: boolean

Values:

- TRUE = power feed voltage is within specification (18 to 32V dc).
- FALSE = power feed is outside specification.

Description:
Reports the health of power feed 1 (nominal 24V dc) to the 9110 processor module in the given slot.

Processor Module C 24V1 Power Feed Health

Direction: input to application from controller

Type: boolean

Values:

- TRUE = power feed voltage is within specification (18 to 32V dc).
- FALSE = power feed is outside specification.

Description:
Reports the health of power feed 1 (nominal 24V dc) to the 9110 processor module in the given slot.

Processor Module A 24V2 Power Feed Health

Direction: input to application from controller

Type: boolean

Values:

- TRUE = power feed voltage is within specification (18 to 32V dc).
- FALSE = power feed is outside specification.

Description:
Reports the health of power feed 2 (nominal 24V dc) to the 9110 processor module in the given slot.

Processor Module B 24V2 Power Feed Health

Direction: input to application from controller

Type: boolean

Values:

- TRUE = power feed voltage is within specification (18 to 32V dc).
- FALSE = power feed is outside specification.

Description:
Reports the health of power feed 2 (nominal 24V dc) to the 9110 processor module in the given slot.

Processor Module C 24V2 Power Feed Health

Direction: input to application from controller

Type: boolean

Values:

- TRUE = power feed voltage is within specification (18 to 32V dc).
- FALSE = power feed is outside specification.

Description:
Reports the health of power feed 2 (nominal 24V dc) to the 9110 processor module in the given slot.

Processor Module A Ready

Direction: input to application from controller

Type: boolean

Values:

- TRUE = The 9110 processor module in the given slot is synchronized (see description)
- FALSE = The processor module is out of synchronization or missing.

Description:
Reports that a processor module within a dual or triple modular redundant configuration is present and is synchronized with one or both of its peers. Reports that a simplex processor module is present.

Processor Module B Ready

Direction: input to application from controller

Type: boolean

Values:

- TRUE = The 9110 processor module in the given slot is synchronized (see description)
- FALSE = The processor module is out of synchronization or missing.

Description:
Reports that a processor module within a dual or triple modular redundant configuration is present and is synchronized with one or both of its peers. Reports that a simplex processor module is present.

Processor Module C Ready

Direction: input to application from controller

Type: boolean

Values:

- TRUE = The 9110 processor module in the given slot is synchronized (see description)
- FALSE = The processor module is out of synchronization or missing.

Description:
Reports that a processor module within a dual or triple modular redundant configuration is present and is synchronized with one or both of its peers. Reports that a simplex processor module is present.

Processor Module A NVRAM Battery Health

Direction: input to application from controller

Type: boolean

Values:

- TRUE = The back-up battery in the 9110 processor module in the given slot is present and its voltage is within acceptable limits.
- FALSE = The voltage of the back-up battery is low or the battery is missing.

Description:
Reports the health status of the back-up battery in a processor module.

Note: The battery voltage is checked at start up, then re-checked every 24 hours (elapsed time).

Processor Module B NVRAM Battery Health

Direction: input to application from controller

Type: boolean

Values:

- TRUE = The back-up battery in the 9110 processor module in the given slot is present and its voltage is within acceptable limits.
- FALSE = The voltage of the back-up battery is low or the battery is missing.

Description:
Reports the health status of the back-up battery in a processor module.

Note: The battery voltage is checked at start up, then re-checked every 24 hours (elapsed time).

Processor Module C NVRAM Battery Health

Direction: input to application from controller

Type: boolean

Values:

- TRUE = The back-up battery in the 9110 processor module in the given slot is present and its voltage is within acceptable limits.
- FALSE = The voltage of the back-up battery is low or the battery is missing.

Description:
Reports the health status of the back-up battery in a processor module.

Note: The battery voltage is checked at start up, then re-checked every 24 hours (elapsed time).

Control Booleans

The variables in the rack of control booleans enable the application to send specific information to the controller.

Unlock All Locked Variables

Direction: output from application to controller

Type: boolean

Values:

- TRUE = Remove all locks.
- FALSE = No effect.
- Default FALSE

Description:
Removes all user locks on input and output variables.

Set System Health Alarm

Direction: [TBD]

Type: boolean

Values:

- TRUE = [TBD]
- FALSE = [TBD]
- Default [TBD]

Description:
[TBD]

RTC Status Variables

The RTC status variables provide information about the controller real-time clock to the application.

RTC Status: Year

Direction: input to application from controller

Type: word

Values:

- 2,000 to 2,399, or 0 (see description)

Description:
Reports the oldest value of real-time clock (RTC) year as voted by every 9110 processor module which is present and synchronized. Only updated if the real-time clock control Boolean RTC Read is set to TRUE. If RTC Read is FALSE, the value will be 0 (zero).

RTC Status: Month

Direction: input to application from controller

Type: word

Values:

- 1 to 12, or 0 (see description)

Description:
Reports the oldest value of real-time clock (RTC) month as voted by every 9110 processor module which is present and synchronized. Only updated if the real-time clock control Boolean RTC Read is set to TRUE. If RTC Read is FALSE, the value will be 0 (zero).

RTC Status: Day of Month

Direction: input to application from controller

Type: word

Values:

- 1 to 31, or 0 (see description)

Description:
Reports the oldest value of real-time clock (RTC) day of the month as voted by every 9110 processor module which is present and synchronized. Only updated if the real-time clock control Boolean RTC Read is set to TRUE. If RTC Read is FALSE, the value will be 0 (zero).

RTC Status: Hours

Direction: input to application from controller

Type: word

Values:

- 0 to 23

Description:
Reports the oldest value of real-time clock (RTC) hours as voted by every 9110 processor module which is present and synchronized. Only updated if the real-time clock control Boolean RTC Read is set to TRUE. If RTC Read is FALSE, the value will be 0 (zero).

RTC Status: Minutes

Direction: input to application from controller

Type: word

Values:

- 0 to 59

Description:
Reports the oldest value of real-time clock (RTC) minutes as voted by every 9110 processor module which is present and synchronized. Only updated if the real-time clock control Boolean RTC Read is set to TRUE. If RTC Read is FALSE, the value will be 0 (zero).

RTC Status: Seconds

Direction: input to application from controller

Type: word

Values:

- 0 to 59

Description:
Reports the oldest value of real-time clock (RTC) seconds as voted by every 9110 processor module which is present and synchronized. Only updated if the real-time clock control Boolean RTC Read is set to TRUE. If RTC Read is FALSE, the value will be 0 (zero).

RTC Status: Milliseconds

Direction: input to application from controller

Type: word

Values:

- 0 to 999

Description:
Reports the oldest value of real-time clock (RTC) milliseconds as voted by every 9110 processor module which is present and synchronized. Only updated if the real-time clock control Boolean RTC Read is set to TRUE. If RTC Read is FALSE, the value will be 0 (zero).

RTC Program Variables

The variables in the rack of RTC program variables specify parts of the date to be written to the real-time clock the next time the RTC control variable **RTC Write** is asserted TRUE.

Note: The values will be written only if the RTC control variable **Year** is TRUE.

RTC Program: Year

Direction: output from application to controller

Type: word

Values:

- 2,000 to 2,399
- Default 0 (zero)

Description:
Specifies the year part of the date to be written to the real-time clock the next time the RTC control variable **RTC Write** is asserted TRUE. The value will be written only if the RTC control variable **Year** is TRUE.

RTC Program: Month

Direction: output from application to controller

Type: word

Values:

- 1 to 12
- Default 0 (zero)

Description:
Specifies the number of the month part of the date to be written to the real-time clock the next time the RTC control variable **RTC Write** is asserted TRUE. The value will be written only if the RTC control variable **Month** is TRUE.

RTC Program: Day of Month

Direction: output from application to controller

Type: word

Values:

- 1 to 31
- Default 0 (zero)

Description:
Specifies the day of the month part of the date to be written to the real-time clock the next time the RTC control variable **RTC Write** is asserted TRUE. The value will be written only if the RTC control variable **Day** is TRUE.

RTC Program: Hours

Direction: output from application to controller

Type: word

Values:

- 0 to 23
- Default 0 (zero)

Description:
Specifies the time of day (in hours) to be written to the real-time clock the next time the RTC control variable **RTC Write** is asserted TRUE. The value will be written only if the RTC control variable **Hours** is TRUE.

RTC Program: Minutes

Direction: output from application to controller

Type: word

Values:

- 0 to 59
- Default 0 (zero)

Description:
Specifies the time of day (in minutes) to be written to the real-time clock the next time the RTC control variable **RTC Write** is asserted TRUE. The value will be written only if the RTC control variable **Minutes** is TRUE.

RTC Program: Seconds

Direction: output from application to controller

Type: word

Values:

- 0 to 59
- Default 0 (zero)

Description:
Specifies the time of day (in seconds) to be written to the real-time clock the next time the RTC control variable **RTC Write** is asserted TRUE. The value will be written only if the RTC control variable **Seconds** is TRUE.

RTC Program: Milliseconds

Direction: output from application to controller

Type: word

Values:

- 0 to 999
- Default 0 (zero)

Description:
Specifies the time of day (in milliseconds) to be written to the real-time clock the next time the RTC control variable **RTC Write** is asserted TRUE. The value will be written only if the RTC control variable **Milliseconds** is TRUE.

RTC Control Variables

The variables in the rack of RTC control variables regulate updates to the real-time clock.

RTC Control: RTC Write

Direction: output from application to controller

Type: boolean

Values:

- TRUE = Applies new values to real-time clock (see description).
- FALSE = No effect.
- Default FALSE.

Description:
Sets new values for the real-time clock. There are six values, all specified by the RTC program control words **Year**, **Month**, **Day**, **Hours**, **Minutes** and **Seconds**. Each value will be set only if its associated RTC control variable (which is a Boolean, and similarly named **Year**, **Month**, **Day**, **Hours**, **Minutes** or **Seconds**) is TRUE.

The change is initiated by the transition from FALSE to TRUE and actioned at the end of the application cycle. The application must hold the TRUE state at least until the end of the cycle for the clock to be updated.

There is no time limit on returning the value from TRUE to FALSE.

Example

Consider this scenario:

The date is 28th October 2008, 8 hours, 12 minutes and 35 seconds

RTC Control RTC Read is TRUE
RTC Control Year, **Month** and **Day of Month** are TRUE
RTC Control Hours, **Minutes** and **Seconds** are TRUE

The RTC status variables will be returned, and the real-time clock will be set, like this:

Year = 2,008
Month = 10
Day = 28
Hours = 8
Minutes = 12
Seconds = 35

RTC Control: RTC Read

Direction: output from application to controller

Type: boolean

Values:

- TRUE = The controller updates RTC status values on each application cycle.
- FALSE = RTC status values are static (do not update).
- Default: [TBD]

Description:
Determines whether the RTC status variables (**RTC status year**, **RTC status month**, **RTC status day of month**, **RTC status hours**, **RTC status minutes** and **RTC status seconds**) will update in real time.

Important Note: All the RTC Status variables must be set to TRUE when the RTC Read variable is set to TRUE, otherwise the RTC value will not be updated and reported.

RTC Control: Year

Direction: output from application to controller

Type: boolean

Values:

- TRUE = RTC program year will be applied by **RTC Write**.
- FALSE = RTC program year will be ignored.
- Default FALSE until an initial value is specified in the application.

Description:
Defines whether the value of the RTC program variable named **Year** should be applied to the real-time clock the next time the RTC control variable named **RTC Write** is set to TRUE.

Note: The RTC program variable is only updated if the RTC control variable RTC Read is set to TRUE and all other RTC Control variables are set to TRUE.

RTC Control: Month

Direction: output from application to controller

Type: boolean

Values:

- TRUE = RTC program month will be applied by **RTC Write**.
- FALSE = RTC program month will be ignored.
- Default FALSE until an initial value is specified in the application.

Description:
Defines whether the value of the RTC program variable named **Month** should be applied to the real-time clock the next time the RTC control variable named **RTC Write** is set to TRUE.

Note: The RTC program variable is only updated if the RTC control variable RTC Read is set to TRUE and all other RTC variables are set to TRUE.

RTC Control: Day of Month

Direction: output from application to controller

Type: boolean

Values:

- TRUE = RTC program day of month will be applied by **RTC Write**.
- FALSE = RTC program day of month will be ignored.
- Default FALSE until an initial value is specified in the application.

Description:
Defines whether the value of the RTC program variable named **Day of Month** should be applied to the real-time clock the next time the RTC control variable named **RTC Write** is set to TRUE.

Note: The RTC program variable is only updated if the RTC control variable RTC Read is set to TRUE and all other RTC Control variables are set to TRUE.

RTC Control: Hours

Direction: output from application to controller

Type: boolean

Values:

- TRUE = RTC program hours will be applied by **RTC Write**.
- FALSE = RTC program hours will be ignored.
- Default FALSE until an initial value is specified in the application.

Description:
Defines whether the value of the RTC program variable named **Hours** should be applied to the real-time clock the next time the RTC control variable named **RTC Write** is set to TRUE.

Note: The RTC program variable is only updated if the RTC control variable RTC Read is set to TRUE and all other RTC Control variables are set to TRUE.

RTC Control: Minutes

Direction: output from application to controller

Type: boolean

Values:

- TRUE = RTC program minutes will be applied by **RTC Write**.
- FALSE = RTC program minutes will be ignored.
- Default FALSE until an initial value is specified in the application.

Description:
Defines whether the value of the RTC program variable named **Minutes** should be applied to the real-time clock the next time the RTC control variable named **RTC Write** is set to TRUE.

Note: The RTC program variable is only updated if the RTC control variable RTC Read is set to TRUE and all other RTC Control variables are set to TRUE.

RTC Control: Seconds

Direction: output from application to controller

Type: boolean

Values:

- TRUE = RTC program seconds will be applied by **RTC Write**.
- FALSE = RTC program seconds will be ignored.
- Default FALSE until an initial value is specified in the application.

Description:
Defines whether the value of the RTC program variable named **Seconds** should be applied to the real-time clock the next time the RTC control variable named **RTC Write** is set to TRUE.

Note: The RTC program variable is only updated if the RTC control variable RTC Read is set to TRUE and all other RTC Control Variables are set to TRUE.

RTC Control: Milliseconds

Direction: output from application to controller

Type: boolean

Values:

- TRUE = RTC program milliseconds will be applied by **RTC Write**.
- FALSE = RTC program milliseconds will be ignored.
- Default FALSE until an initial value is specified in the application.

Description:
Defines whether the value of the RTC program variable named **Milliseconds** should be applied to the real-time clock the next time the RTC control variable named **RTC Write** is set to TRUE.

Note: The RTC program variable is only updated if the RTC control variable RTC Read is set to TRUE and all other RTC Control Variables are set to TRUE.

This page intentionally left blank

Chapter 5

Configuring the Controller I/O

This chapter describes the configuration process for defining the controller I/O hardware in the AADvance Workbench.

In This Chapter

About Configuring I/O Modules

I/O modules are configured by selecting an I/O bus in the **Equipment** tree view and then assigning a module to an empty I/O slot. You can configure single modules or two/three modules to form a redundant group to match the arrangement of your hardware.

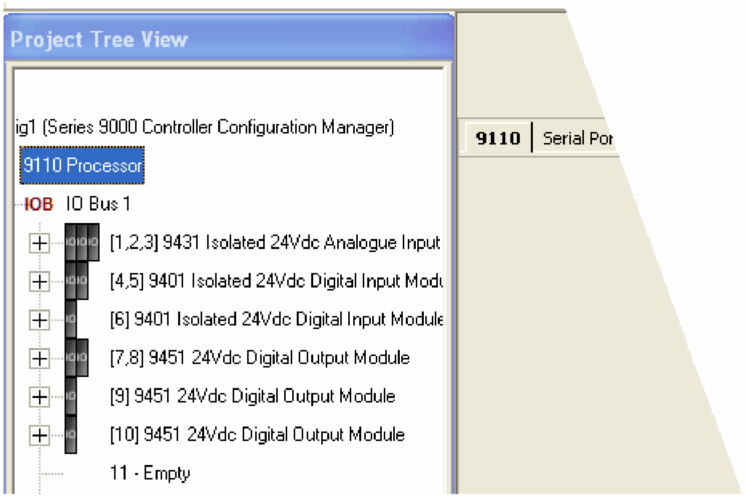
If you choose to insert one module it will be allocated to the slot you have selected. If you choose to insert more than one module, they will automatically be allocated to adjacent slots. To change the configuration you can clear a slot or move a module to another slot.

Use this process to configure I/O modules:

1) Assign I/O modules to the **IO Bus 1** or **IO Bus 2** slots.
2) Set the process safety time for the I/O modules.
3) Configure the I/O module status variables.
4) Configure the I/O module channel variables.

Note: The procedures assume that you have already set up all the variables in the **Dictionary**. If you create a new variable during this process , you will be prompted to store it in the **Dictionary**.

Example I/O Slot Configuration


Project Tree View

ig1 (Series 9000 Controller Configuration Manager)

9110 Processor

IOB IO Bus 1

[1,2,3] 9431 Isolated 24Vdc Analogue Input

[4,5] 9401 Isolated 24Vdc Digital Input Modu

[6] 9401 Isolated 24Vdc Digital Input Module

[7,8] 9451 24Vdc Digital Output Module

[9] 9451 24Vdc Digital Output Module

[10] 9451 24Vdc Digital Output Module

11 - Empty

9110 | Serial Por

In the example illustrated, the modules have been configured as follows:

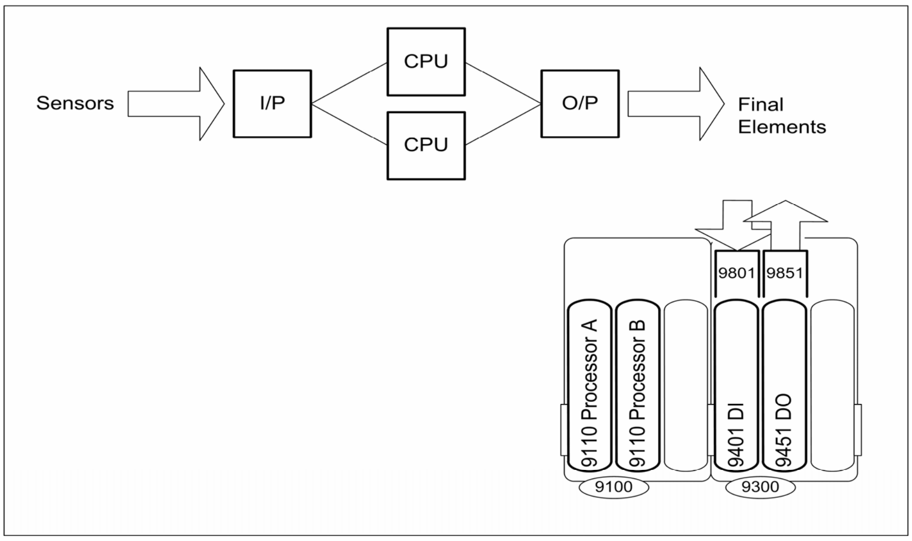
- A redundant group of analogue input modules has been configured in the first three slots.
- Two digital input modules are in the next two slots.
- There is a single digital input module.
- There is a group of two digital output modules.
- Two single digital output modules.

Defining the I/O Hardware Architecture

The I/O hardware architecture is the physical arrangement of the I/O modules in the AADvance controller. To define the I/O hardware architecture in the AADvance Workbench you assign the modules to empty slot numbers on the processor I/O buses. Use the Project Tree View to do this. If desired, you can clear an I/O slot or move an assigned module to a different slot.

Example Controller Configuration

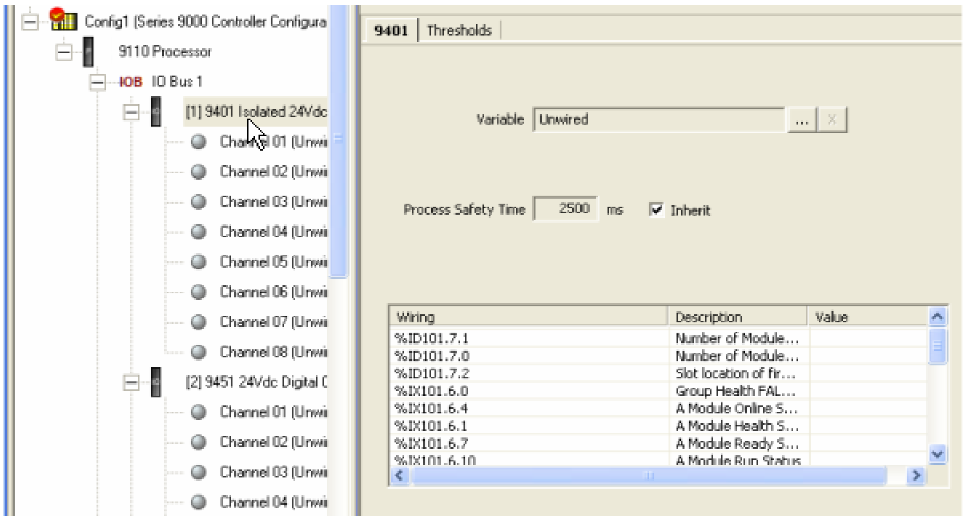
This example controller has two 9110 processor modules and supports 8 digital inputs and 8 digital outputs.



Note: The '9801' and '9851' illustrated are simplex termination assemblies for the I/O modules and provide the connections for the field elements.

This controller has the following physical layout: the two I/O modules are installed to the right of the processor base unit, which is **IO Bus 1**. The 9401 is installed in the first I/O base unit connector, which is slot 1; the 9451 is installed adjacent to the 9401 in the next connector on the I/O base unit, which is slot 2.

You now have to configure the same arrangement in the project tree, connect variables to monitor module status information and I/O data. Use the Project Tree View within the AADvance Workbench to assign the I/O modules to empty slot numbers on the processor IOB IO Bus 1 or IO Bus 2.



Variable: Unwired

Process Safety Time: 2500 ms [✓ Inherit]

Wiring	Description	Value
%ID101.7.1	Number of Module...	
%ID101.7.0	Number of Module...	
%ID101.7.2	Slot location of fir...	
%IX101.6.0	Group Health FAL...	
%IX101.6.4	A Module Online S...	
%IX101.6.1	A Module Health S...	
%IX101.6.7	A Module Ready S...	
%IX101.6.10	A Module Run Status	

The slot and bus numbers must be the same as the actual physical position of the installed modules. Therefore, for this example you would allocate I/O modules as follows:

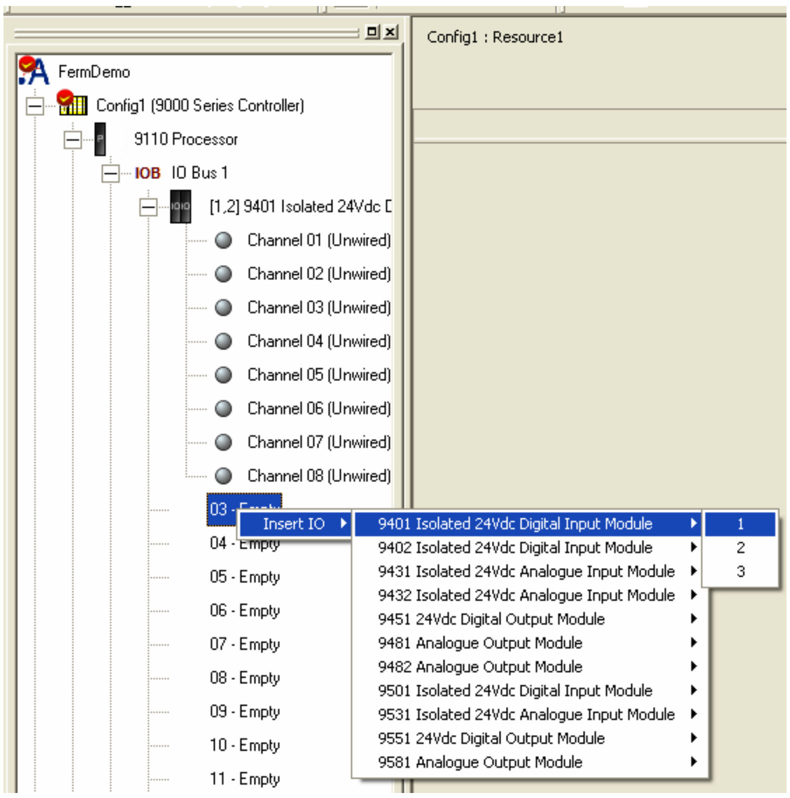
- A 9401 module to the empty slot 1 on IO Bus 1
- A 9451 module to the empty slot 2 on IO Bus 1.

Assign I/O Modules to I/O Bus Slots

If you are assigning a single module you can assign the module to any empty I/O bus slot. If you are creating a redundant group you have to find two or three consecutive empty slots and assign the module to the first empty slot in the group.

Note: If required, you can use the AADvance Workbench to move configured modules to other slots to create a series of adjacent empty slots. Remember to move the actual modules in the controller to the changed slots.

To assign an I/O module do the following:


FermDemo
Config1 (9000 Series Controller)
9110 Processor
IOB IO Bus 1
[1,2] 9401 Isolated 24Vdc D
Channel 01 (Unwired)
Channel 02 (Unwired)
Channel 03 (Unwired)
Channel 04 (Unwired)
Channel 05 (Unwired)
Channel 06 (Unwired)
Channel 07 (Unwired)
Channel 08 (Unwired)
03 - Empty
04 - Empty
05 - Empty
06 - Empty
07 - Empty
08 - Empty
09 - Empty
10 - Empty
11 - Empty
Config1 : Resource1
Insert IO
9401 Isolated 24Vdc Digital Input Module
9402 Isolated 24Vdc Digital Input Module
9431 Isolated 24Vdc Analogue Input Module
9432 Isolated 24Vdc Analogue Input Module
9451 24Vdc Digital Output Module
9481 Analogue Output Module
9482 Analogue Output Module
9501 Isolated 24Vdc Digital Input Module
9531 Isolated 24Vdc Analogue Input Module
9551 24Vdc Digital Output Module
9581 Analogue Output Module
1
2
3

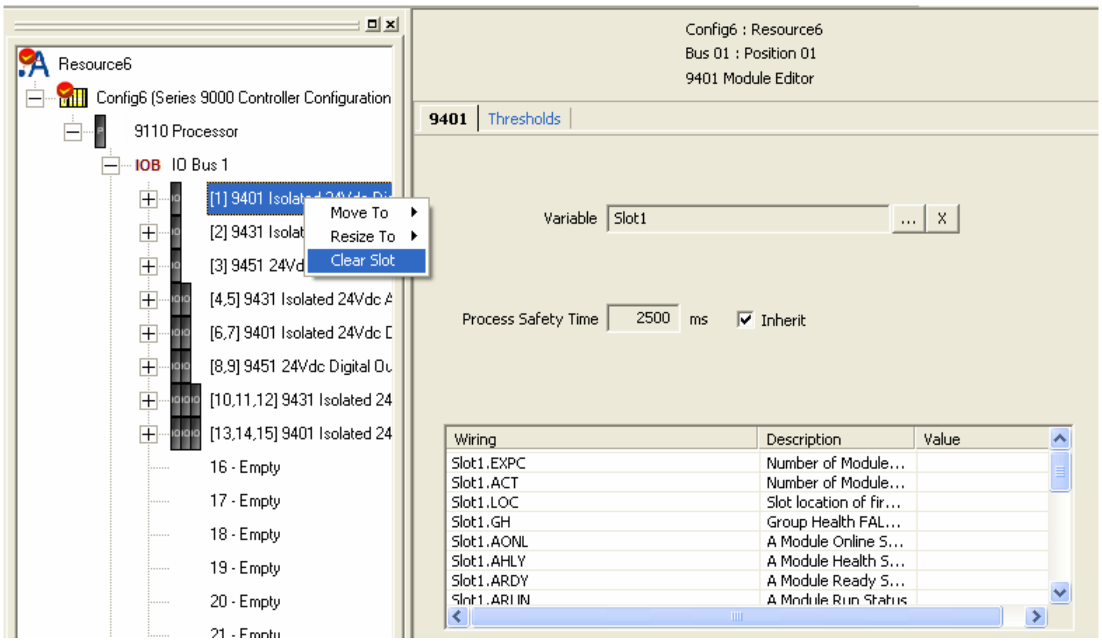
1) Select the **Equipment** tab.
2) Expand the **IO Bus 1**.
3) At an empty slot right-click to select **Insert IO**. Move the cursor to the right to select from the choice of available modules (empty slots position 16 - 24 shown in bold).

4) Select the required module, then move the cursor to the right and select the number of modules you require.

5) Repeat this for all the modules you want to configure.

Clear an I/O Bus Slot

To clear an I/O bus slot do the following:



Resource6
Config6 (Series 9000 Controller Configuration
9110 Processor
IOB IO Bus 1
[1] 9401 Isolated 24Vdc Di
[2] 9431 Isolat
[3] 9451 24Vd
[4,5] 9431 Isolated 24Vdc A
[6,7] 9401 Isolated 24Vdc D
[8,9] 9451 24Vdc Digital Ou
[10,11,12] 9431 Isolated 24
[13,14,15] 9401 Isolated 24
16 - Empty
17 - Empty
18 - Empty
19 - Empty
20 - Empty
21 - Empty

Config6 : Resource6
Bus 01 : Position 01
9401 Module Editor

Variable: Slot1

Process Safety Time: 2500 ms ☑ Inherit

Wiring	Description	Value
Slot1.EXPC	Number of Module…	
Slot1.ACT	Number of Module…	
Slot1.LOC	Slot location of fir…	
Slot1.GH	Group Health FAL…	
Slot1.AONL	A Module Online S…	
Slot1.AHLY	A Module Health S…	
Slot1.ARDY	A Module Ready S…	
Slot1.ARUN	A Module Run Status	

1) Select the **Equipment** tab.

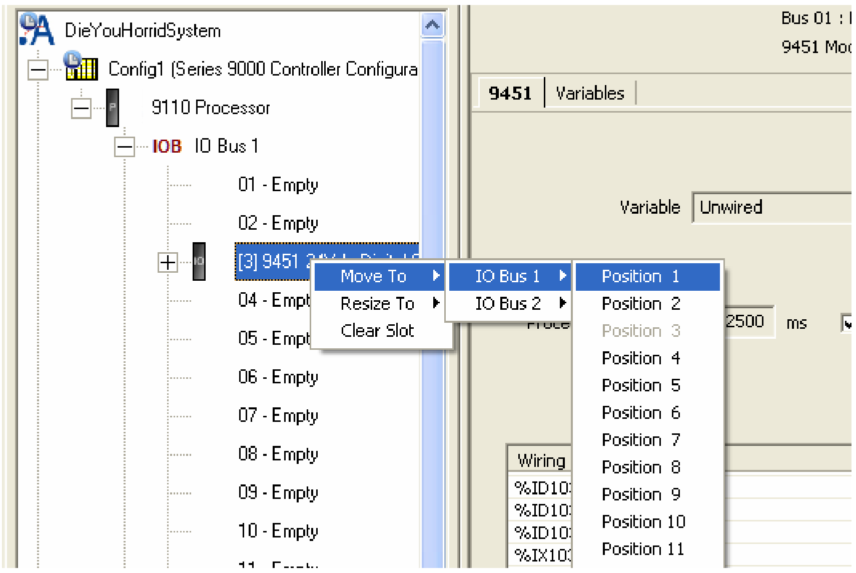
2) Right-click to select **Clear Slot**.

- The module will be removed from the slot.
- The slot will now display as **Empty** ready to be re-assigned a module.

Note: The channel variable wiring will automatically be unwired.

Move a Module to a Different Slot

To move an assigned module to a different slot do the following:



1) Select the **Equipment** tab.
2) Right-click to select the **Move To** option.
3) Move the cursor to the right and select the **IO Bus 1 Position** you want to move to.

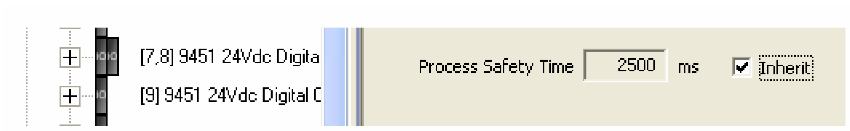
  ‣ The module is automatically re-assigned to the selected slot.

Note: The channel variable wiring will move with the module to the new slot and is automatically renumbered.

Configure the I/O Module Process Safety Time

When you configure the process safety time for an I/O module, you can choose to inherit the top-level value set for the processor or specify a value for the I/O module.

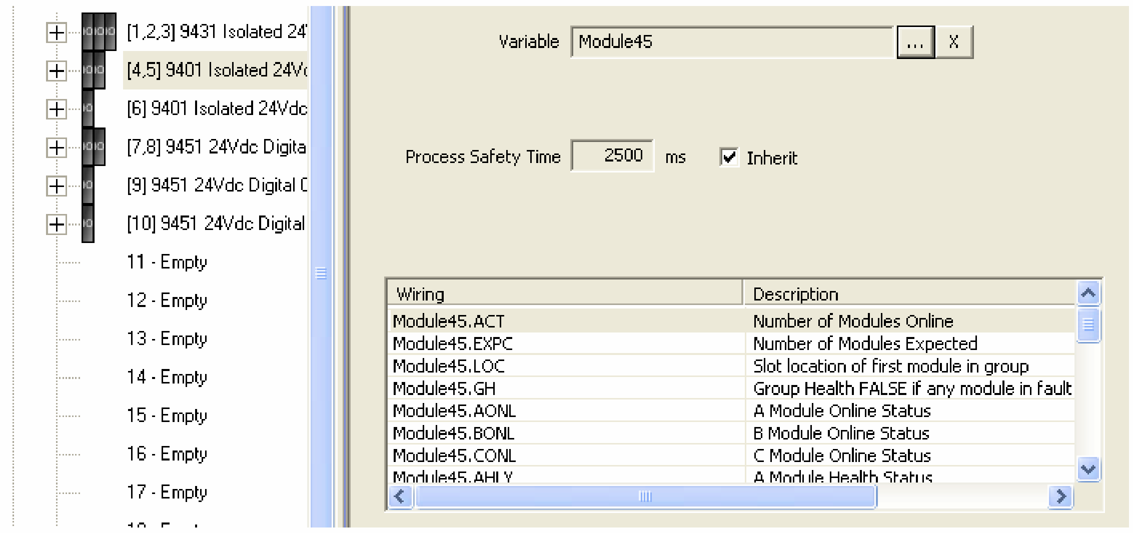
To define the I/O module process safety time do the following:



1) Select a module from the I/O Bus.
2) Put a tick in the **Inherit** box to inherit the top-level **Process Safety Time**; or
3) De-select the **Inherit** box and enter a value into the **Process Safety Time** field.

Wire Status Variables to I/O Modules

You can wire a variable to an I/O module so the application can receive status information from the module. The AADvance Workbench provides a structure (**T9K_TA_GROUP_STATUS**) for module status information. To wire a status variable to an I/O module do the following:



Variable: Module45

Process Safety Time: 2500 ms ✓ Inherit

Wiring	Description
Module45.ACT	Number of Modules Online
Module45.EXPC	Number of Modules Expected
Module45.LOC	Slot location of first module in group
Module45.GH	Group Health FALSE if any module in fault
Module45.AONL	A Module Online Status
Module45.BONL	B Module Online Status
Module45.CONL	C Module Online Status
Module45.AHLY	A Module Health Status

1) Declare a variable in Dictionary. Use the type **T9K_TA_GROUP_STATUS** and make sure that the direction is set to **input**.

2) Select a module from the I/O bus.

   - The **unwired** term appears in the **Variable** field.

3) Click the ... button adjacent to the **Variable** field.

   - The variables dialog box opens.

4) Select the variable you declared in the Dictionary.

   - The variable is displayed in the **Variable** field.
   - The status variables are automatically assigned and appear in the **Wiring** column with the description in the **Description** column.

5) Repeat steps 1 to 4 for other I/O modules.

T9K_TA_GROUP_STATUS (I/O Module Status Information)

The data structure for module status information (**T9K_TA_GROUP_STATUS**) provides the elements detailed in the table.

Note: The controller interrogates an I/O module (designated 'X' in the table) according to the physical arrangement of the module and its position in a group. A simplex module is designated as module A; a duplex module as A or B and a triplicated module as A, B or C.

Table 9: Structure for I/O Module Status Data

Identifier	Type	Description	Remarks
<tagname>.EXPC	INT	Modules expected	Reports the number of modules that are defined in the configuration for the group (1, 2 or 3)
<tagname>.ACT	INT	Modules on-line	Reports the number of modules in a group that are installed, powered, locked and communicating over the I/O bus (1, 2 or 3)
<tagname>.LOC	INT	Slot location	Reports the slot number of the left-most module position for a group, irrespective of whether a module is physically located in a slot (1 to 24) (†)
<tagname>.GH	BOOL	Group health	Reports the general health status of all modules in a group TRUE: all modules are healthy FALSE: one or more modules in the group is on-line and reporting a fault
<tagname>.XONL	BOOL	On-line status	Reports the on-line status of module X TRUE: the module is installed, powered, locked and is communicating over the I/O bus, otherwise FALSE
<tagname>.XHLY	BOOL	Health status	Reports the general health of module X TRUE: the module is on-line and has no faults, otherwise FALSE
<tagname>.XRDY	BOOL	Ready status	Reports the ready status of module X TRUE: the module is on-line and ready to report channel values, otherwise FALSE
<tagname>.XRUN	BOOL	Run status	Reports the run status of module X TRUE: the module is on-line and reporting channel values, or requires manual intervention (pressing the Fault Reset button) before values can be reported, otherwise FALSE

<tagname>.XSDN	BOOL	Shutdown status	Reports that module X requires manual intervention (pressing the Fault Reset button) before values can be reported TRUE: the module needs manual intervention
<tagname>.XPOS	INT	Position	Reports the slot number of module X (1 to 24) (†)

Note: (†) Slots are numbered 1 to 24 on both buses; the slot location (**.LOC**) and position (**XPOS**) do not identify the bus.

About Configuring I/O Channels

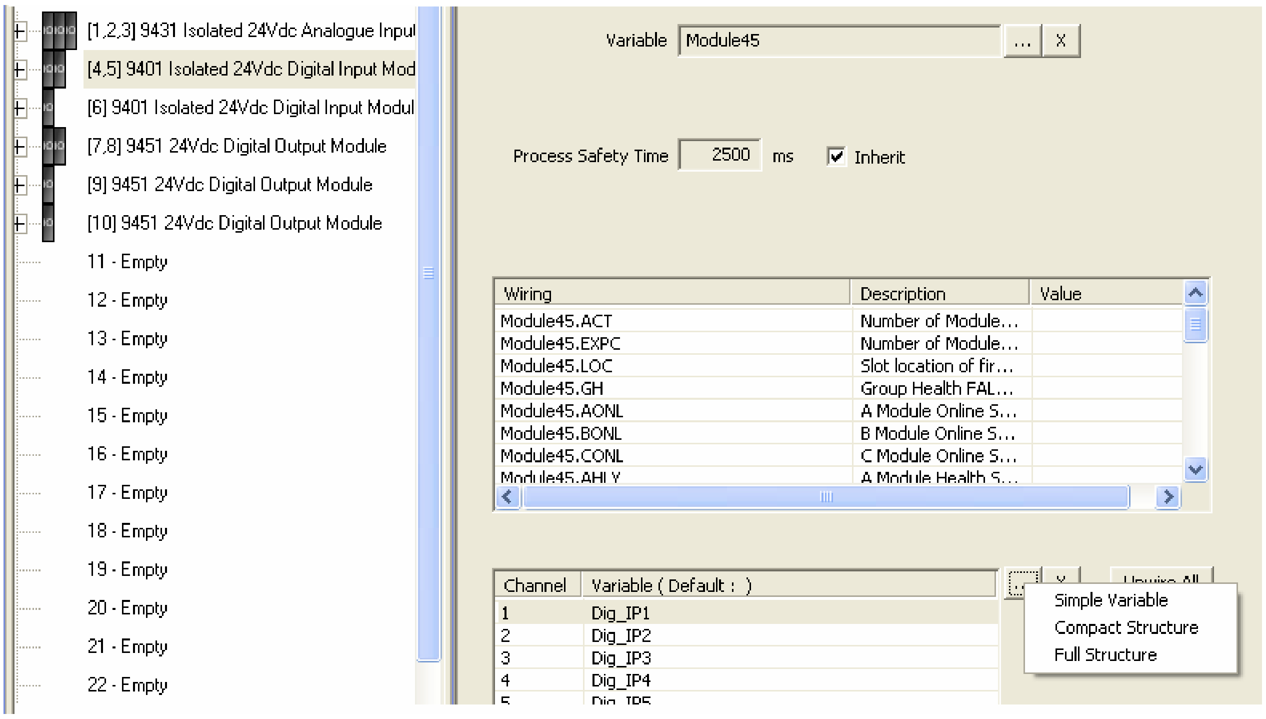
The AADvance Workbench provides a set of variable structures to wire to I/O channels. When you declare the I/O channels variables using your own tagnames you can declare one of two types of structure (compact and full); alternatively, the primary variable can be assigned directly to the base variable type.

The AADvance Workbench will automatically generate a set of variable elements with the same tagname; thus depending on the chosen structure, the system automatically wires a set of I/O variables to the channels.

The syntax for a structure variable is <**tagname**>.XX where XX represents the reporting element of the variable; for example, <**tagname**>.DI is a Boolean that reports the digital input state for a channel.

Wire Variables to Digital Input Channels

To wire variables to digital input channels do the following:



[1,2,3] 9431 Isolated 24Vdc Analogue Input
[4,5] 9401 Isolated 24Vdc Digital Input Mod
[6] 9401 Isolated 24Vdc Digital Input Modul
[7,8] 9451 24Vdc Digital Output Module
[9] 9451 24Vdc Digital Output Module
[10] 9451 24Vdc Digital Output Module
11 - Empty
12 - Empty
13 - Empty
14 - Empty
15 - Empty
16 - Empty
17 - Empty
18 - Empty
19 - Empty
20 - Empty
21 - Empty
22 - Empty

Variable: Module45

Process Safety Time: 2500 ms ☑ Inherit

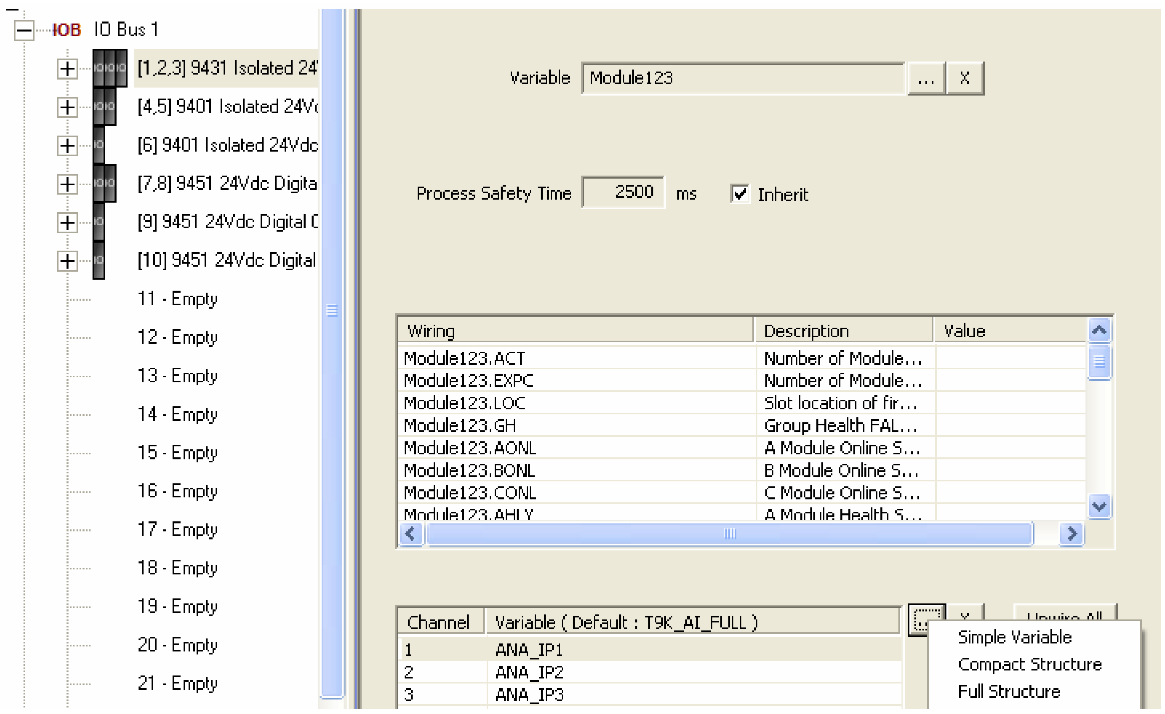
Wiring	Description	Value
Module45.ACT	Number of Module...	
Module45.EXPC	Number of Module...	
Module45.LOC	Slot location of fir...	
Module45.GH	Group Health FAL...	
Module45.AONL	A Module Online S...	
Module45.BONL	B Module Online S...	
Module45.CONL	C Module Online S...	
Module45.AHLY	A Module Health S...	

Channel	Variable (Default :)
1	Dig_IP1
2	Dig_IP2
3	Dig_IP3
4	Dig_IP4
5	Dig_IP5

1) Select a digital input module on the I/O Bus.
   - The module status variable <**tagname**> that you assigned will appear in the **Variable** field.
2) Select the channel that you want to wire to a variable.
3) Click the ... button adjacent to the **Channel Variable** fields.
4) Choose a data structure from the three options displayed: **Simple, Compact, Full.**
   - The **Select Variables** dialog box opens.
5) Select a named structure, click **OK.**
6) Repeat steps 2 to 5 for each channel you want to wire.

Wire Variables to Analogue Input Channels

To wire variables to analogue input channels do the following:



IOB IO Bus 1
- [1,2,3] 9431 Isolated 24
- [4,5] 9401 Isolated 24V
- [6] 9401 Isolated 24Vdc
- [7,8] 9451 24Vdc Digita
- [9] 9451 24Vdc Digital
- [10] 9451 24Vdc Digital
- 11 - Empty
- 12 - Empty
- 13 - Empty
- 14 - Empty
- 15 - Empty
- 16 - Empty
- 17 - Empty
- 18 - Empty
- 19 - Empty
- 20 - Empty
- 21 - Empty

Variable: Module123

Process Safety Time: 2500 ms [✓ Inherit]

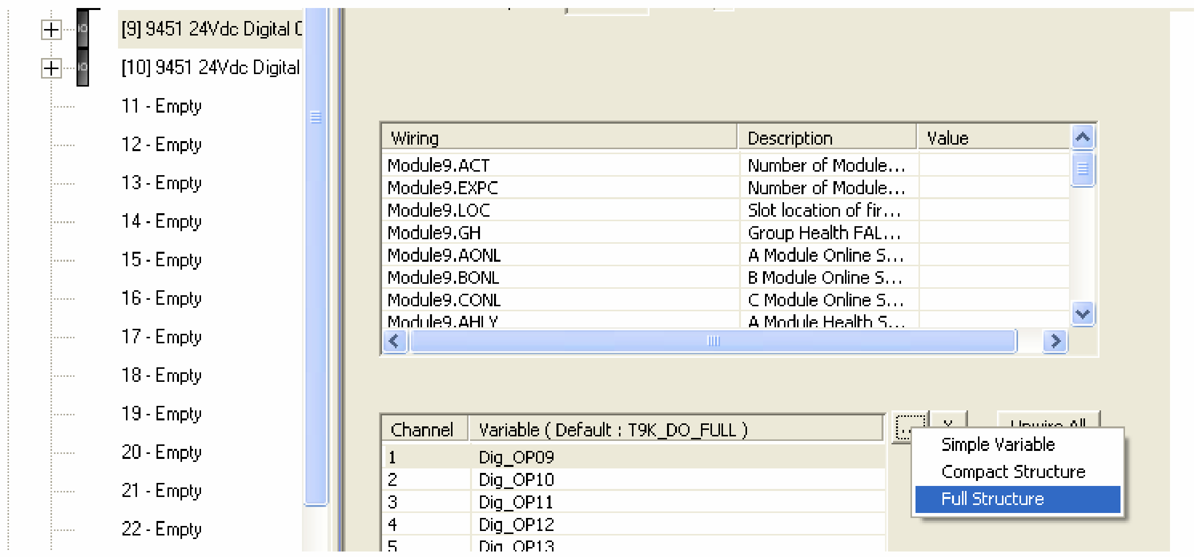
Wiring	Description	Value
Module123.ACT	Number of Module...	
Module123.EXPC	Number of Module...	
Module123.LOC	Slot location of fir...	
Module123.GH	Group Health FAL...	
Module123.AONL	A Module Online S...	
Module123.BONL	B Module Online S...	
Module123.CONL	C Module Online S...	
Module123.AHLY	A Module Health S...	

Channel	Variable (Default : T9K_AI_FULL)
1	ANA_IP1
2	ANA_IP2
3	ANA_IP3

1) Select an analogue input module on the I/O bus.
   - The module status variable <**tagname**> that you assigned will appear in the **Variable** field.
2) Select the channel that you want to wire to a variable.
3) Click the button next to the channel variable fields.
4) Choose a data structure from the three options displayed: **Simple, Compact, Full.**
   - The **Select Variables** dialog box is displayed.
5) Select a named structure, click **OK.**
6) Repeat steps 2 to 5 for each channel.

Wire Variables to Digital Output Channels

To wire variables to digital output channels do the following:



[9] 9451 24Vdc Digital
[10] 9451 24Vdc Digital
11 - Empty
12 - Empty
13 - Empty
14 - Empty
15 - Empty
16 - Empty
17 - Empty
18 - Empty
19 - Empty
20 - Empty
21 - Empty
22 - Empty

Wiring	Description	Value
Module9.ACT	Number of Module...	
Module9.EXPC	Number of Module...	
Module9.LOC	Slot location of fir...	
Module9.GH	Group Health FAL...	
Module9.AONL	A Module Online S...	
Module9.BONL	B Module Online S...	
Module9.CONL	C Module Online S...	
Module9.AHLY	A Module Health S...	

Channel	Variable (Default : T9K_DO_FULL)
1	Dig_OP09
2	Dig_OP10
3	Dig_OP11
4	Dig_OP12
5	Dig_OP13

1) Select a digital output module on the I/O bus.
   - The module status variable <**tagname**> appears in the **Variable** field.
2) Select the channel that you want to wire to a variable.
3) Click the ... button next to the **Channel Variable** fields.
4) Choose a data structure from the three options displayed: **Simple, Compact, Full.**
   - The **Select Variables** dialog box is displayed.
5) Select a named structure, click **OK**.
6) Repeat steps 2 to 5 for each channel.

Configuring Digital Inputs

You can wire digital input channels to the following variable type and data structures:

- **BOOL** (the <variable_name> gives the input state)
- **TK9_DI_Compact** (provides three elements)
- **TK9_DI_Full** (six elements)

The structures provide additional information about the input, such as line fault status and discrepancy status. You can also define custom thresholds for digital inputs.

TK9_DI_COMPACT and TK9_DI_FULL (Digital Inputs)

The two data structures for digital input channels (**TK9_DI_COMPACT** and **TK9_DI_FULL**) provide the elements detailed in the tables.

Table 10: TK9_DI_COMPACT Structure for Digital Inputs

Identifier	Type	Description	Remarks
<tagname>.DI	BOOL	Input state	TRUE: input voltage above threshold T6 FALSE: input voltage below threshold T5
<tagname>.LF	BOOL	Line fault	TRUE: input voltage above threshold T8; between T5 and T4; or below T1 FALSE: input voltage between thresholds T2 and T3; or between T6 and T7
<tagname>.DIS	BOOL	Discrepancy	TRUE: there is a discrepancy in voltage greater than 20% between the channels of two or three modules in a redundant configuration (†)

Table 11: TK9_DI_FULL Structure for Digital Inputs

Identifier	Type	Description	Remarks
<tagname>.DI	BOOL	Input state	TRUE: input voltage above threshold T6 FALSE: input voltage below threshold T5
<tagname>.LF	BOOL	Line fault	TRUE: input voltage above threshold T8; between T5 and T4; or below T1 FALSE: input voltage between thresholds T2 and T3; or between T6 and T7
<tagname>.DIS	BOOL	Discrepancy	TRUE: there is a discrepancy in voltage greater than 8% (of 24V) between the channels of two or three modules in a redundant configuration (†)
<tagname>.CF	BOOL	Channel fault	TRUE: module diagnostics detect a fault in the channel electronics or firmware (state = 7)
<tagname>.V	UINT	Voltage	Reports the channel voltage in units of millivolts and with an accuracy of ± 500mV (††)
<tagname>.STA	USINT	State	Reports a state value for the channel: 1 = open circuit 2 = de-energized 3 = indeterminate 4 = energized 5 = short-circuit 6 = over voltage 7 = faulted

Note: (†) Discrepancy can only be reported TRUE when two or three modules are active in a group. (††) The voltage element cannot report values below 0mV.

Faulted State for Digital Inputs

A digital input channel is faulted (the state reports a value of 7) when the channel is incapable of reporting a voltage within a safety accuracy specification of 10% of the full scale measurement of the 24V dc supply (2.4V).

When the state reports the value 7, then the following 'safe' values are reported by the other variables:

- Input State = FALSE
- Line Fault = TRUE
- Discrepancy = TRUE
- Channel Fault = TRUE
- Voltage = 0mV

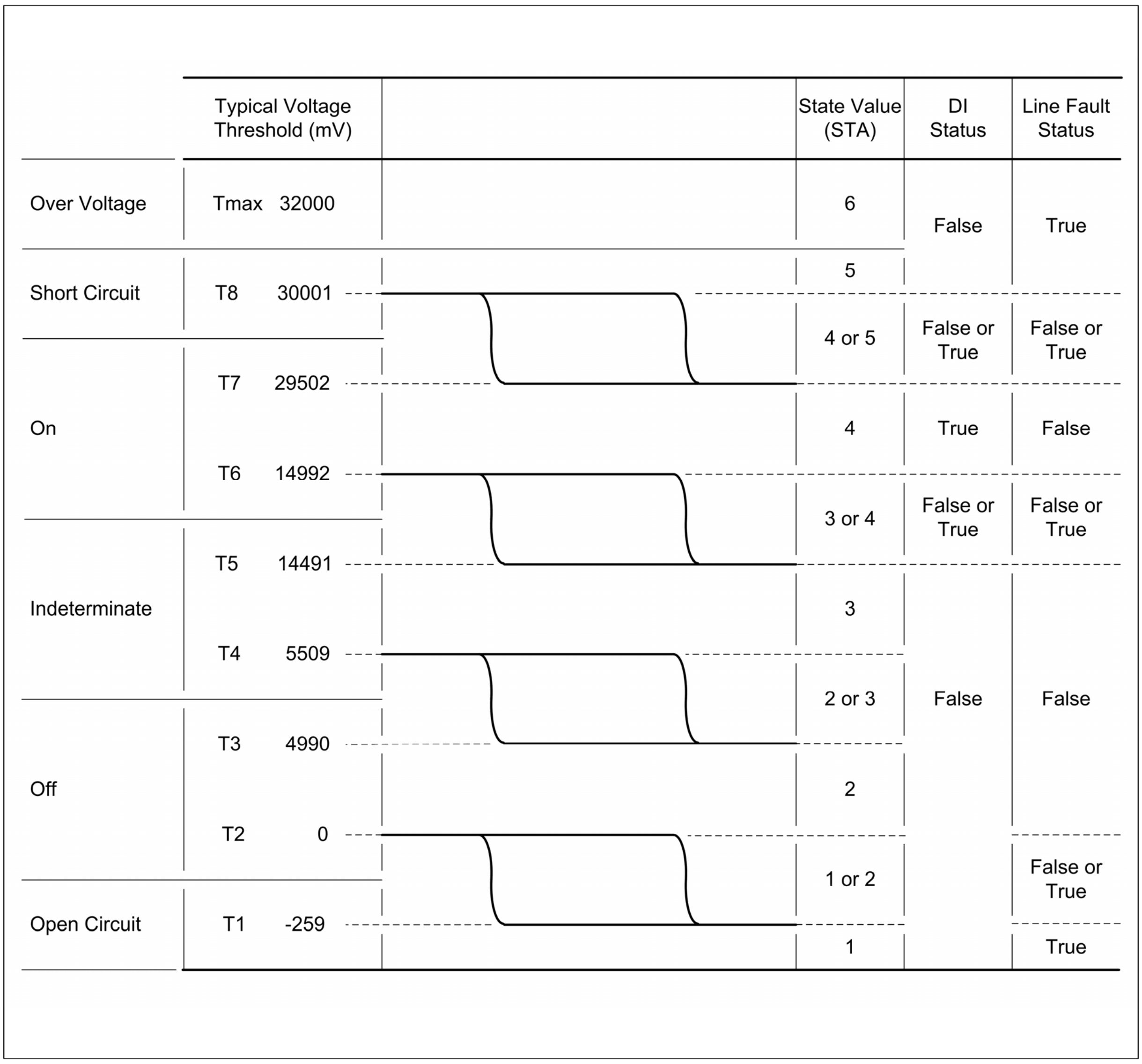
About Threshold Values for Digital Inputs

The module determines the channel state and the line fault status by comparing the channel input voltage with defined threshold values. You can define your own threshold values or use a set of default values. The values you choose for the module are inherited by each channel; you can define different thresholds for individual channels later.

An indeterminate region is defined between the closed and open status to allow for marginal faults in the external wiring or sensor.

Note: When the system is operational you should change these values only through an on-line update.

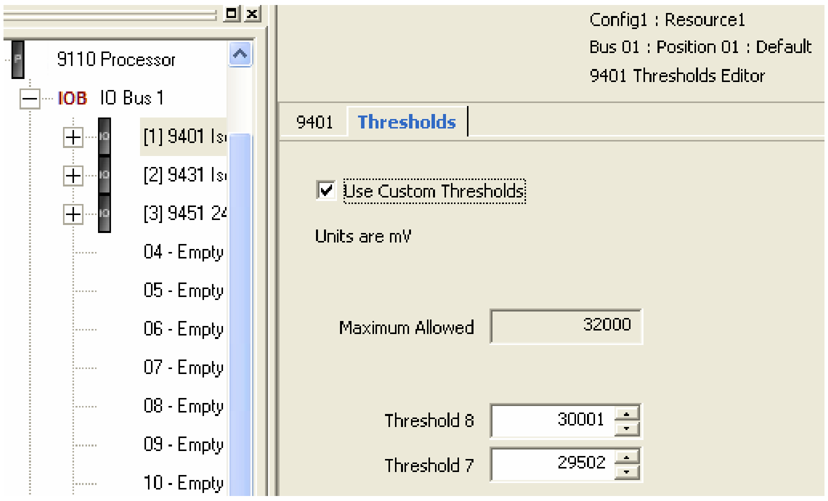
The AADvance controller provides hysteresis on the thresholds for increasing and decreasing values to prevent chatter. The Workbench updates the reporting values during every application cycle.



	Typical Voltage Threshold (mV)	State Value (STA)	DI Status	Line Fault Status
Over Voltage	Tmax 32000	6	False	True
		5		
Short Circuit	T8 30001	4 or 5	False or True	False or True
On	T7 29502	4	True	False
	T6 14992	3 or 4	False or True	False or True
Indeterminate	T5 14491	3	False	False
	T4 5509	2 or 3		
Off	T3 4990	2		
	T2 0	1 or 2		False or True
Open Circuit	T1 -259	1		True

Define Thresholds for a Digital Input Module

To define your own threshold values do the following:


9110 Processor
IOB IO Bus 1
[1] 9401 Is
[2] 9431 Is
[3] 9451 24
04 - Empty
05 - Empty
06 - Empty
07 - Empty
08 - Empty
09 - Empty
10 - Empty

Config1 : Resource1
Bus 01 : Position 01 : Default
9401 Thresholds Editor

9401 | Thresholds

☑ Use Custom Thresholds

Units are mV

Maximum Allowed: 32000

Threshold 8: 30001

Threshold 7: 29502

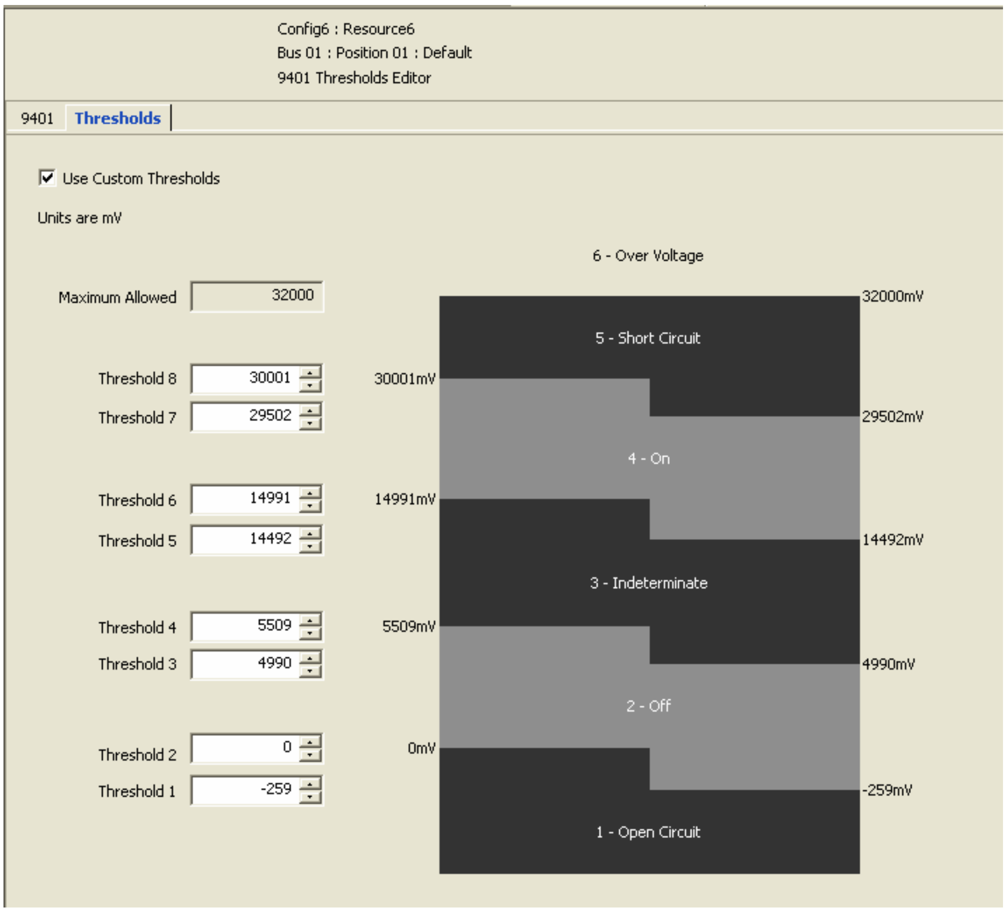
1) Select the **Thresholds** tab on the module editor.
   - A set of default values is shown in the threshold fields.
2) To enter your own values select the **Use Custom Thresholds** box, enter your own values in the threshold fields, click **Apply**.
3) To restore the default values, Click **Default** then de-select the **Use Custom Thresholds**, click **Apply**.

Default Thresholds for Digital Inputs

The default threshold values for digital inputs are for a standard (non-line monitored) 24V dc digital input channel.

The default values are given in the table.

Default Threshold Values for the 9401 Digital Input Module



Config6 : Resource6
Bus 01 : Position 01 : Default
9401 Thresholds Editor

9401 **Thresholds**

☑ Use Custom Thresholds

Units are mV

Maximum Allowed	32000
Threshold 8	30001
Threshold 7	29502
Threshold 6	14991
Threshold 5	14492
Threshold 4	5509
Threshold 3	4990
Threshold 2	0
Threshold 1	-259

Configuring Analogue Inputs

You can wire analogue input channels to the following variable type and data structures:

- REAL (the <variable_name> gives a floating-point value representing 4 to 20mA)
- TK9_AI_Compact (provides three elements)
- TK9_AI_Full (six elements)

The structures provide additional information about the input, such as discrepancy status. You can also configure analogue inputs to operate with HART devices, and define custom thresholds.

TK9_AI_COMPACT and TK9_AI_FULL (Analogue Inputs)

The two data structures for analogue inputs (**TK9_AI_COMPACT** and **TK9_AI_FULL)** provide the elements detailed in the tables.

Table 12: TK9_AI_COMPACT Structure for Analogue Inputs

Identifier	Type	Description	Remarks
<tagname>.PV	REAL	PV	Process Value. A scaled, floating-point value representing the analogue loop current. Default scaling factor is 0 to 100% representing 4 to 20mA
<tagname>.CNT	INT	Raw count	A count representing the current on the channel in units of 1/256mA 0 represents 0mA 5,120 represents 20mA Accurate to within ± 13 counts, equivalent to ± 0.05mA
<tagname>.DIS	BOOL	Discrepancy	TRUE: there is a discrepancy in current greater than 2% between the channels of two or three modules in a redundant configuration (†)

Table 13: TK9_AI_FULL Structure for Analogue Inputs

Identifier	Type	Description	Remarks
<tagname>.PV	REAL	PV	Process Value. A scaled, floating-point value representing the analogue loop current. Default scaling factor is 0 to 100% representing 4 to 20mA

<tagname>.CNT	INT	Raw count	A count representing the current on the channel in units of 1/256mA 0 represents 0mA; 5,120 represents 20mA Accurate to within ± 13 counts, equivalent to ± 0.05mA
<tagname>.LF	BOOL	Line fault	TRUE: state (.STA) is 1, 5, 6 or 7 FALSE: state (.STA) is 2, 3 or 4
<tagname>.DIS	BOOL	Discrepancy	TRUE: there is a discrepancy in current greater than 2% between the channels of two or three modules in a redundant configuration (†)
<tagname>.CF	BOOL	Channel fault	TRUE: module diagnostics detect a fault in the channel electronics or firmware (state = 7)
<tagname>.STA	USINT	State	Reports a state value for the channel: 1 = open circuit 2 = transmitter fault (low) 3 = normal 4 = transmitter fault (high) 5 = short-circuit 6 = over range 7 = faulted

Note: (†) Discrepancy can only be reported TRUE when two or three modules are active in a group.

Faulted State for Analogue Inputs

An analogue input channel is faulted (the state reports a value of 7) when the channel is incapable of reporting a count within a safety accuracy specification of 1% of the full scale measurement range of 5,120 (51 counts, 0.2mA).

When the state reports the value 7 then the following 'safe' values are reported by the other variables:

- Process Value = a calculated value based on a Count value of 0
- Line Fault = TRUE
- Discrepancy = TRUE
- Channel Fault = TRUE
- Count = 0

About HART

The AADvance controller is the first Rockwell Automation controller to include integrated support for HART (**Highway Addressable Remote Transducer**) communications. There is no need for separate HART interfaces. The AADvance Workbench supports HART on analogue input channels; the system implements revision 5 of the HART specification.

The application program can use HART information to monitor and respond to device conditions. You can also use HART to carry out maintenance and diagnostic functions on field devices.

The AADvance Workbench provides a dedicated data structure, **T9K_AI_HART**, for application variables that will use the HART functionality. The structure provides the following information:

- The loop current
- Four pre-defined dynamic variables, and their associated units
- Communication and device status information

Using HART

Make sure that your HART devices support HART command 0 ('**read unique ID**') and HART command 3 ('**read current** and four pre-defined dynamic variables'). The controller uses these commands to communicate with the HART devices.

The HART loop current variable (available within **T9K_AI_HART**) is in addition to the channel variable for the 4 to 20mA loop. You can use the HART loop current variable for diagnostic checks, for example to compare with the value on the 4 to 20mA loop and react if there is a discrepancy; do not use the HART loop current variable for a functional safety application.

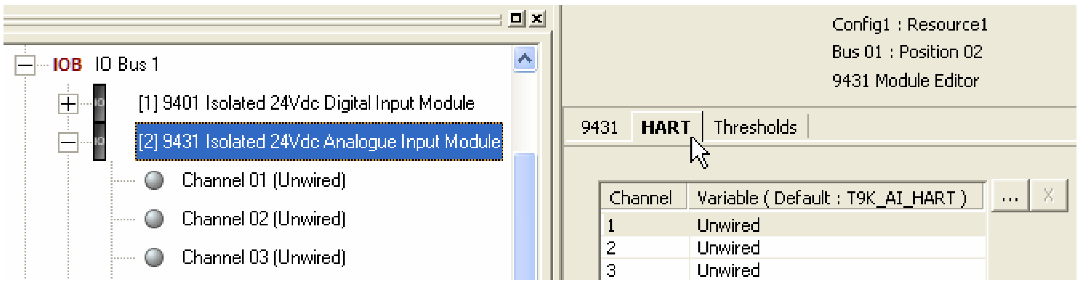
For more details of HART see the HART Application Guide, created by the HART Communication Foundation, and their detailed HART specifications. You can download these documents from www.hartcomm.org.

Configure Analogue Inputs for HART

You configure input channels for HART within the AADvance Workbench. Do the following:

1) Create **HART** variables within the Dictionary; set the type to **T9K_AI_HART**. You need one variable for each HART device.

2) Go to the **Equipment** Tree View and select the analogue input module. Click on the **HART** tab.


IOB IO Bus 1
[1] 9401 Isolated 24Vdc Digital Input Module
[2] 9431 Isolated 24Vdc Analogue Input Module
Channel 01 (Unwired)
Channel 02 (Unwired)
Channel 03 (Unwired)

Config1 : Resource1
Bus 01 : Position 02
9431 Module Editor

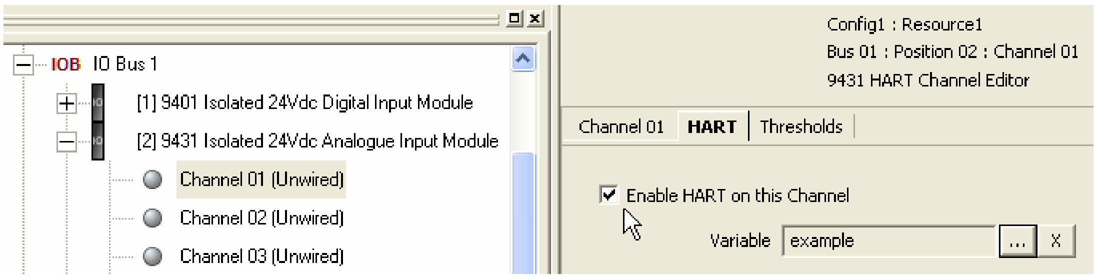
9431 | HART | Thresholds

Channel	Variable (Default : T9K_AI_HART)
1	Unwired
2	Unwired
3	Unwired

3) Select a **Channel**, click ... .

4) Choose a variable from the Dictionary list, click **OK**.

5) Return to the **Equipment** Tree View, select the channel and click on the **HART** tab; then put a tick in the box labeled **Enable HART on this Channel**, click **Apply**.


IOB IO Bus 1
[1] 9401 Isolated 24Vdc Digital Input Module
[2] 9431 Isolated 24Vdc Analogue Input Module
Channel 01 (Unwired)
Channel 02 (Unwired)
Channel 03 (Unwired)

Config1 : Resource1
Bus 01 : Position 02 : Channel 01
9431 HART Channel Editor

Channel 01 | HART | Thresholds

[✓] Enable HART on this Channel
Variable: example

6) Repeat this procedure for the other inputs that will use HART-enabled devices.

T9K_AI_HART (HART Input Devices)

The data structure for **HART** devices (**T9K_AI_HART**) provides the elements detailed in the table.

Table 14: HART Data Structure

Identifier	Type	Description	Remarks
<tagname>.I	REAL	Current (mA)	
<tagname>.V1	REAL	Primary variable	
<tagname>.U1	BYTE	Primary variable units code	
<tagname>.V2	REAL	Second variable	
<tagname>.U2	BYTE	Second variable units code	
<tagname>.V3	REAL	Third variable	
<tagname>.U3	BYTE	Third variable units code	
<tagname>.V4	REAL	Fourth variable	
<tagname>.U4	BYTE	Fourth variable units code	
<tagname>.COMMS	BOOL	Communication status	TRUE: [TBD] FALSE: [TBD]

<tagname>.DEVICE	BYTE	Device status (†)	Bit 7: field device malfunction Bit 6: configuration changed Bit 5: cold start Bit 4: more status available Bit 3: analogue output current fixed Bit 2: analogue output saturated Bit 1: non-primary variable out of limits Bit 0: primary variable out of limits

Note: (†) The device status byte mimics the **HART** field device status. Appendix E of the **HART** Application Guide gives details.

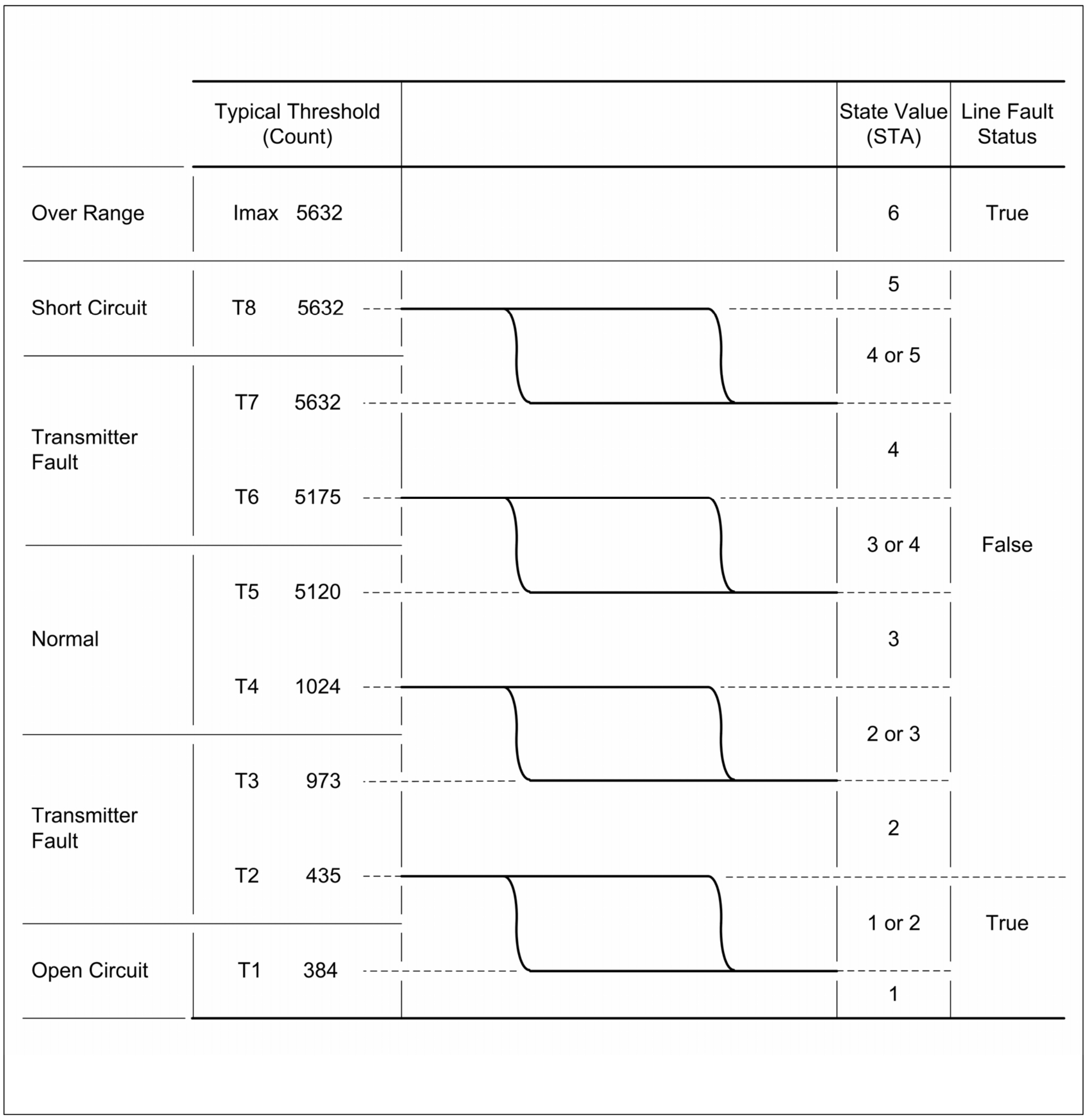
About Threshold Values for Analogue Inputs

The module determines the channel state and the line fault status by comparing the channel input current with defined threshold values. You can define your own threshold values or use a set of default values. The values you choose for the module are inherited by each channel; you can define different thresholds for individual channels later.

Thresholds are specified in counts, with 0 (zero) being 0mA, 1,024 being 4mA, and 5,120 being 20mA.

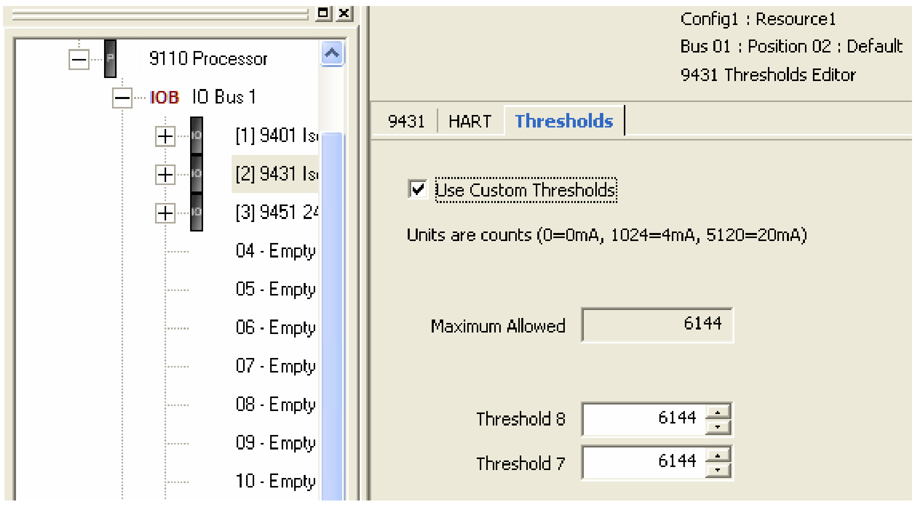
Note: When the system is operational you should change these values only using an on-line update.

The AADvance controller provides hysteresis on the thresholds for increasing and decreasing values to prevent chatter. The Workbench updates the reporting values during every application cycle.



Define Thresholds for an Analogue Input Module

To define your own threshold values do the following:



Config1 : Resource1
Bus 01 : Position 02 : Default
9431 Thresholds Editor

[✓] Use Custom Thresholds

Units are counts (0=0mA, 1024=4mA, 5120=20mA)

Maximum Allowed 6144

Threshold 8 6144

Threshold 7 6144

1) Select the **Thresholds** tab on the module editor.
   - A set of default values is shown in the threshold fields.
2) To enter your own values select the **Use Custom Thresholds** box, enter your own values in the threshold fields, click **Apply**.

Note: You can enter the values in counts (the default units) or in milliamps. To specify a value in milliamps, append 'mA' to the value; the AADvance Workbench will convert it into counts.

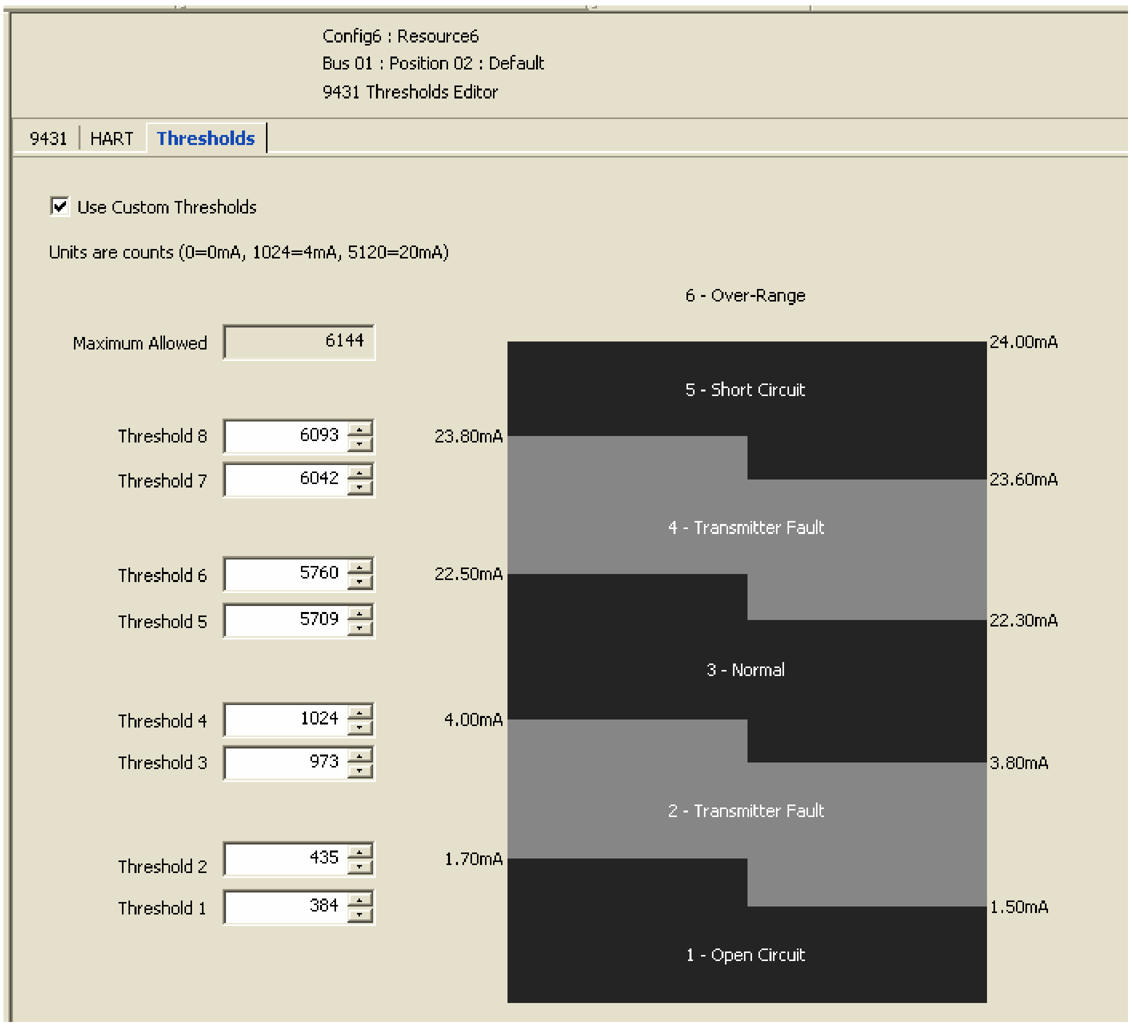
3) To restore the default values, Click **Default** then de-select the **Use Custom Thresholds** box, click **Apply**.

Default Thresholds for Analogue Inputs

The default threshold values for analogue inputs are for a standard (non-line monitored) 24V dc analogue input channel.

The default values are given in the table.

Table 15: Default Threshold Values for the 9431 Analogue Input Module



Config6 : Resource6
Bus 01 : Position 02 : Default
9431 Thresholds Editor

[✓] Use Custom Thresholds

Units are counts (0=0mA, 1024=4mA, 5120=20mA)

Maximum Allowed	6144
Threshold 8	6093
Threshold 7	6042
Threshold 6	5760
Threshold 5	5709
Threshold 4	1024
Threshold 3	973
Threshold 2	435
Threshold 1	384

Configuring Digital Outputs

You can wire digital output channels to the following variable type and data structures:

- **BOOL** (the <variable_name> gives the commanded state)
- **TK9_DO_Compact (provides three elements)**
- **TK9_DO_Full** (seven elements)

The structures provide additional information about the output, such as line fault status and discrepancy status.

Note: The controller writes its digital outputs once per application cycle; the digital output variables are also updated once per application cycle.

TK9_DO_COMPACT and TK9_DO_FULL (Digital Outputs)

The data structures for digital inputs (**TK9_DO_COMPACT** and **TK9_DO_FULL**) provide the elements detailed in the tables.

Table 16: TK9_DO_COMPACT Structure for Digital Outputs

Identifier	Type	Description	Remarks
<tagname>.DOP	BOOL	Input state	The commanded state to be passed to the output channel Set to TRUE to energize Set to FALSE to de-energize
<tagname>.LF	BOOL	Line fault	TRUE: no field supply is present, no load is connected, or a short circuit is detected
<tagname>.DIS	BOOL	Discrepancy	TRUE: there is a discrepancy in current greater than 1% between the channels of two modules in a redundant configuration (†)

Table 17: TK9_DO_FULL Structure for Digital Outputs

Identifier	Type	Description	Remarks
<tagname>.DOP	BOOL	Input state	The commanded state to be passed to the output channel Set to TRUE to energize Set to FALSE to de-energize
<tagname>.LF	BOOL	Line fault	TRUE: no field supply is present, no load is connected, or a short circuit is detected

<tagname>.DIS	BOOL	Discrepancy	TRUE: there is a discrepancy in current greater than 1% between the channels of two modules in a redundant configuration (†)
<tagname>.CF	BOOL	Channel fault	TRUE: module diagnostics detect a fault in the channel electronics or firmware (state = 7)
<tagname>.V	UINT	Voltage	Reports the channel voltage at the output terminals, in units of millivolts and with an accuracy of ± 500mV (††)
<tagname>.I	INT	Current	Reports the current for the channel in milliamps and with an accuracy of ± 2mA and ± 10% of measurement
<tagname>.STA	USINT	Channel state	Reports a state value for the channel: 1 = no vfield 2 = de-energized 3 = no load 4 = energized 5 = short-circuit 6 = field fault 7 = faulted

Note: (†) Discrepancy can only be reported TRUE when two modules are active in a group. (††) The voltage element cannot report values below 0mV.

The State Variable for Digital Outputs

The state variable for a digital output is an unsigned integer with a value from 1 to 7 representing the following:

1 = no-vfield: the field supply voltage is at or below 18V dc for that channel.

Note: When the state variable is 1, the field voltage (<tagname.V>) is reported as 0mV.

2 = de-energized: the commanded state is FALSE and the channel is de-energized.

3 = no-load: the controller cannot detect a load connected to the channel field wiring, or the load is below the minimum required channel load of 10mA when commanded TRUE.

4 = energized: the commanded state is TRUE and the channel is energized.

5 = short-circuit: the controller has detected a short-circuit condition, irrespective of the channel drive state.

6 = field fault: an external source is driving the channel to an energized state or a voltage greater than 18V dc, irrespective of the channel drive state.

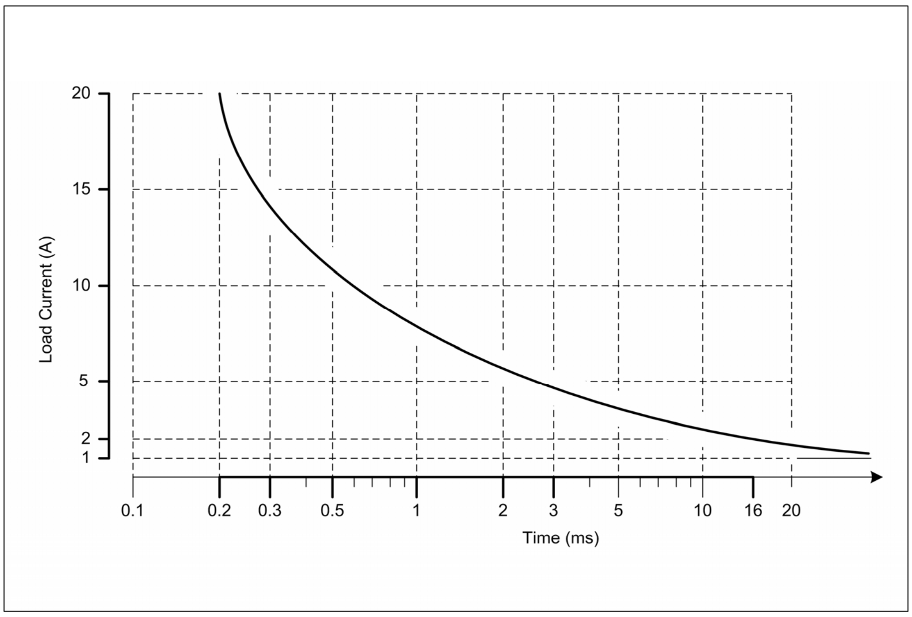
7 = faulted.

Overcurrent Protection for Digital Outputs

The AADvance controller has three mechanisms to protect its digital output channels:

- Inrush current protection
- Short circuit protection for energized channels
- Short circuit protection for de-energized channels

The controller tolerates inrush currents so that its digital outputs can energize capacitive loads without causing the controller to report a short circuit. The illustration shows the characteristics of the maximum load currents that the controller will tolerate when a digital output is commanded on. If the load current enters the region above the curve on the graph, the controller applies its inrush current protection.



After allowing for inrush, the controller engages its short circuit protection for an energized channel when the loop current reaches 2A.

- Short circuit detection on an energized channel is immediate and the channel is de-energized. The controller reports the condition until the short circuit is cleared.
- When the short circuit is removed, the channel will re-energize. The short circuit report is then cleared by pressing the fault reset button on the 9110 processor module or by setting the commanded state is set to FALSE.

The controller checks de-energized digital output channels for potential short circuits. Periodically, the controller partially turns on each de-energized output in turn and measures the loop current. If the loop current shows a loop resistance of less than approximately 10Ω, the controller reports a short circuit.

Faulted State for Digital Outputs

A digital output channel is faulted (the state reports a value of 7) when normal operation or diagnostics tests have identified a specific fault condition. A single identified fault condition thus results in a state value of 7.

When the state reports the value 7 then the following 'safe' values are reported by the other variables:

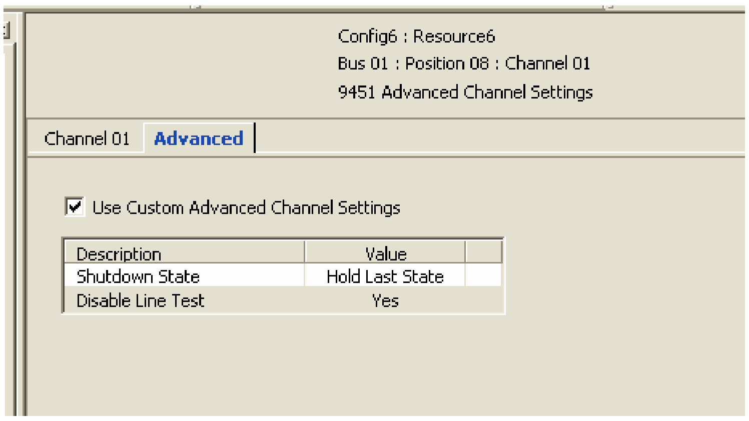
- Line Fault = TRUE
- Discrepancy = TRUE
- Channel Fault = TRUE
- Voltage = 0mV
- Current = 0mA

Configure Advanced Channel Settings for Digital Outputs

The AADvance Workbench provides advanced settings for individual digital output channels:

- You can specify a shutdown state for an output; this defines how the output will behave when its parent 9451 digital output module is in a shutdown mode.
- You can disable the line test feature for an output; this disables detection of a no-load condition.

To configure the advanced channel settings do the following:


Config6 : Resource6
Bus 01 : Position 08 : Channel 01
9451 Advanced Channel Settings
Channel 01
Advanced

☑ Use Custom Advanced Channel Settings

Description	Value
Shutdown State	Hold Last State
Disable Line Test	Yes

1) Select the slot with the digital output module.
   - The module status variable name <tagname> that you assigned appears in the **Variable** field.
2) Select the channel to configure and click the **Advanced** tab.
   - The **9451 Advanced Channel Settings** dialog box opens.
3) Put a tick in the box labelled **Use Custom Advanced Channel Settings.**
4) Put a tick in the box beside the **Shutdown State** field or the **Disable Line Test** field.
5) Choose the advanced channel settings from the drop down options.

Digital Output Advanced Channel Settings

Each output channel from the 9451 digital output module supports the set of control parameters detailed in the table.

Table 18: Digital Output Control Parameters

Description	Value(s)	Default	Remarks
Shutdown State	Off, Hold Last State (†)	None specified	'Off' de-energizes the output during a shutdown 'Hold Last State' forces the output to remain in its last commanded state, during a shutdown
Disable Line Test	Yes, No	None specified	'Yes' disables reporting of the status variable (STA) state 3; in a no-load condition, the Channel LED will not go amber 'No' is equivalent to the default setting, which enables the line test

Note: (†) The option labelled 'Default' does nothing. Do not choose this option.

Disabling the Line Test for a Digital Output

The 9451 digital output module checks for a no-load condition on each output. The AADvance Workbench refers to the check as the 'Line Test'.

A no-load condition occurs when the controller cannot detect a load connected to the field wiring, or the load current is below 20mA when the output is commanded TRUE. You can disable the check for a no-load condition, for example if you want to connect a low load to an output, or if the output is unused and you do not want to fit a dummy load.

The module reports a no-load condition by setting the state variable (<tagname>.STA) to the value 3, and by setting the channel LED to amber. If you disable the line test, then assuming there are no other faults present, the state variable will continue to show 2 or 4 (depending on the commanded value) instead of 3, and the channel LED will show off or green instead of amber.

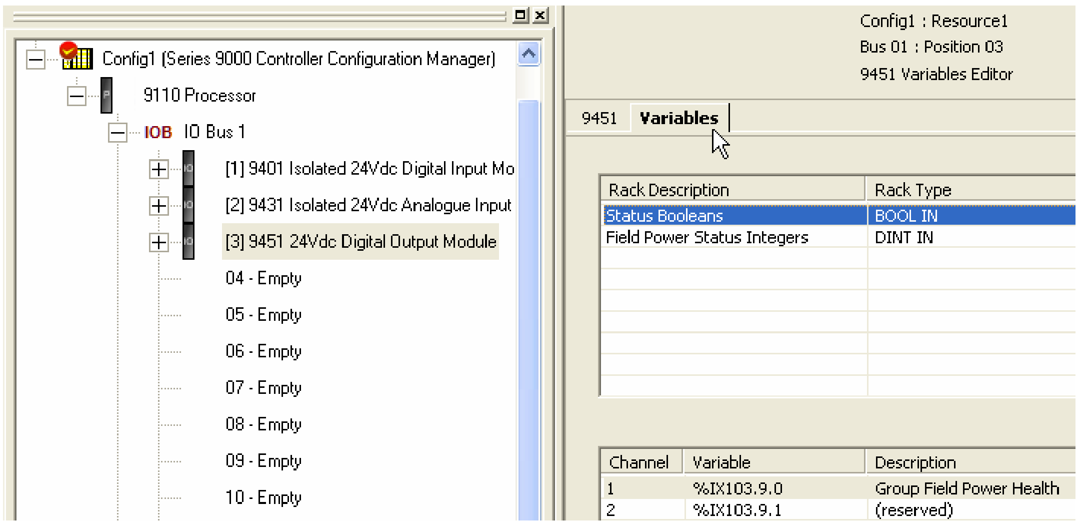
Note: There are other scenarios (such as no field voltage) that still result in an amber LED, even if you disable the line test.

About Status Variables for Digital Output Modules

The 9451 digital output module provides a number of status variables that are available to the application. The 9451 Variables Editor presents the variables in two collections, which it calls 'racks': **Status Booleans**, and **Power Status Integers.**

Wire Status Variables to a Digital Output Module

To wire a status variable to a digital output module do the following:



Rack Description	Rack Type
Status Booleans	BOOL IN
Field Power Status Integers	DINT IN

Channel	Variable	Description
1	%IX103.9.0	Group Field Power Health
2	%IX103.9.1	(reserved)

1) Navigate to the digital output module in the equipment tree view.
2) Select the **Variables** tab of the **9451 Module Editor**.
   - The **9451 Variables Editor** dialog box opens.
3) Select a rack. The editor displays a list of associated variables.

Note: The status variables are for modules, not channels. The column headed 'Channel' shows an index for the variables; it does not relate to individual digital outputs.

4) Select a variable.
5) Click the ... button. The **Select Variable** dialog box opens.
6) From the list select an application variable to wire to the status variable, click **OK**.
7) Repeat for each subsequent variable to be wired.
8) Return to the **9451 Variables Editor** and click **Apply**. The variable will now be wired.

Unwire Status Variables from a Digital Output Module

To disconnect a status variable from a digital output module do the following:

1) Select the **Variables** tab of the **9451 Module Editor**.

- The 9451 Variables Editor dialog box opens.

2) Select the relevant rack.

   - The editor displays a list of associated variables.

3) Select the variable to be unwired, click the **X** button.

4) Click **Apply**.

   - The variable will be unwired.

Note: Select the **Unwire All** button and click **Apply** to disconnect all of the wired variables in the rack.

Status Booleans

The variables in the rack of status booleans provide information to the application about the field power supplies to a group of digital output modules.

Group Field Power Health

Direction: input to application from controller

Type: boolean

Values:

- TRUE = all field power supplies for all active digital output modules in the group are within the range 18V to 32V dc inclusive
- FALSE = one or more field power supplies to an active module is less than 18V dc or greater than 32V dc.

Description:
Provides a top level indication of the health of field power supplies to active digital output modules.

Note: The controller incorporates a 0.5V hysteresis on these thresholds to prevent chatter. The controller will declare a fault when a supply falls below 18V, but will not clear the fault until the supply rises to 18.5V. Similarly the controller will declare a fault when a supply exceeds 32V, but will not clear the fault until the supply falls below 31.5V.

Field Power Status Integers

The variables in the rack of field power status integers (all DINT) provide information to the application about the field power supplies to a group of digital output modules.

Group Field Power Current

Direction: input to application from controller

Type: **DINT**

Values:

- 0 to 8,000mA or greater (limited by capacity of DINT variable)

Description:
Reports the total current that all the active digital output modules in a group are drawing from the field power supply. Accuracy is ± 10%.

A Module Field Power Voltage 1

Direction: input to application from controller

Type: **DINT**

Values:

- 0 to 48,000mV or greater (limited by capacity of DINT variable)

Description:
Reports the voltage from the field power supply, for the specified module and field power input. Accuracy is ± 500mV.

A Module Field Power Voltage 2

Direction: input to application from controller

Type: **DINT**

Values:

- 0 to 48,000mV or greater (limited by capacity of DINT variable)

Description:
Reports the voltage from the field power supply, for the specified module and field power input. Accuracy is ± 500mV.

B Module Field Power Voltage 1

Direction: input to application from controller

Type: **DINT**

Values:

- 0 to 48,000mV or greater (limited by capacity of DINT variable)

Description:
Reports the voltage from the field power supply, for the specified module and field power input. Accuracy is ± 500mV.

B Module Field Power Voltage 2

Direction: input to application from controller

Type: **DINT**

Values:

- 0 to 48,000mV or greater (limited by capacity of DINT variable)

Description:
Reports the voltage from the field power supply, for the specified module and field power input. Accuracy is ± 500mV.