Advantys STB

Standard CANopen Network Interface Module Applications Guide

8/2009

31003684.06

www.schneider-electric.com

Schneider Electric

Schneider Electric assumes no responsibility for any errors that may appear in this document. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

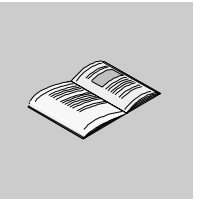
All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

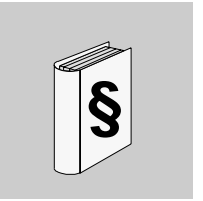
Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

Table of Contents



Safety Information



Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a Danger or Warning safety label indicates that an electrical hazard exists, which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER

DANGER indicates an imminently hazardous situation which, if not avoided, **will result in** death or serious injury.

⚠ WARNING

WARNING indicates a potentially hazardous situation which, if not avoided, **can result in** death or serious injury.

⚠ CAUTION
CAUTION indicates a potentially hazardous situation which, if not avoided, **can result in** minor or moderate injury.

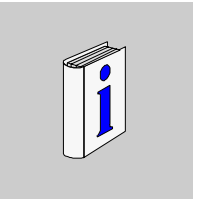
CAUTION
CAUTION, used without the safety alert symbol, indicates a potentially hazardous situation which, if not avoided, **can result in** equipment damage.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and the installation, and has received safety training to recognize and avoid the hazards involved.

About the Book



At a Glance

Document Scope

This guide describes the specific functionality of the STB NCO 2212, the Advantys STB standard network interface module to CANopen. To assist you with setting up your Advantys STB island on a CANopen network, extensive, real-world CANopen application examples are included. These instructions assume the reader has a working familiarity with the CANopen fieldbus protocol.

This guide includes the following information about the STB NCO 2212:

- role in a CANopen network
- role as the gateway to Advantys STB island
- external and internal interfaces
- flash memory and removable memory
- integrated power supply
- auto-configuration
- saving configuration data
- island bus scanner functionality
- data exchange between the island and the master
- diagnostic messages
- specifications

Validity Note

This document is valid for Advantys 4.5 or later.

Related Documents

Title of Documentation	Reference Number
Advantys STB Analog I/O Modules Reference Guide	31007715 (E), 31007716 (F), 31007717 (G), 31007718 (S), 31007719 (I)

Advantys STB Discrete I/O Modules Reference Guide	31007720 (E), 31007721 (F), 31007722 (G), 31007723 (S), 31007724 (I)
Advantys STB Counter Modules Reference Guide	31007725 (E), 31007726 (F), 31007727 (G), 31007728 (S), 31007729 (I)
Advantys STB Special Modules Reference Guide	31007730 (E), 31007731 (F), 31007732 (G), 31007733 (S), 31007734 (I)
Advantys STB System Planning and Installation Guide	31002947 (E), 31002948 (F), 31002949 (G), 31002950 (S), 31002951 (I)
Advantys STB Configuration Software Quick Start User Guide	31002962 (E), 31002963 (F), 31002964 (G), 31002965 (S), 31002966 (I)
Advantys STB Reflex Actions Reference Guide	31004635 (E), 31004636 (F), 31004637 (G), 31004638 (S), 31004639 (I)

You can download these technical publications and other technical information from our website at www.schneider-electric.com.

User Comments

We welcome your comments about this document. You can reach us by e-mail at techcomm@schneider-electric.com.

Introduction

1

Introduction

This chapter describes the STB NCO 2212 standard network interface module and its roles on both the island bus and a CANopen network.

The chapter begins with an introduction of the NIM and a discussion of its role as the gateway to the Advantys STB island. There is a brief overview of the island itself, followed by a description of the major characteristics of the CANopen fieldbus protocol.

What's in this Chapter?

This chapter contains the following topics:

What Is a Network Interface Module?

Purpose

Every island requires a network interface module (NIM) in the leftmost location of the primary segment. Physically, the NIM is the first (leftmost) module on the island bus. Functionally, it is the gateway to the island bus. That is, all communications to and from the island bus pass through the NIM. The NIM also has an integrated power supply that provides logic power to the island modules.

The Fieldbus Network

An island bus is a node of distributed I/O on an open fieldbus network, and the NIM is the island's interface to that network. The NIM supports data transfers over the fieldbus network between the island and the fieldbus master.

The physical design of the NIM makes it compatible with both an Advantys STB island and your specific fieldbus master. Whereas the fieldbus connector on each NIM type may differ, the location on the module front panel is essentially the same.

Communications Roles

Communications capabilities provided on a standard NIM include:

Function	Role
data exchange	The NIM manages the exchange of input and output data between the island and the fieldbus master. Input data, stored in native island bus format, is converted to a fieldbus-specific format that can be read by the fieldbus master. Output data written to the NIM by the master is sent across the island bus to update the output modules and is automatically reformatted.
configuration services	Custom services can be performed by the Advantys configuration software. These services include changing the operating parameters of the I/O modules, fine-tuning island bus performance, and configuring reflex actions. The Advantys Configuration Software runs on a computer attached to the NIM's CFG interface *(see page 35)*. (For NIMs with Ethernet port connectivity, you can also connect to the Ethernet port.)
human-machine interface (HMI) operations	A serial Modbus HMI panel can be configured as an input and/or output device on the island. As an input device, it can write data that can be received by the fieldbus master; as an output device, it can receive updated data from the fieldbus master. The HMI can also monitor island status, data, and diagnostic information. The HMI panel must be attached to the NIM's CFG port.

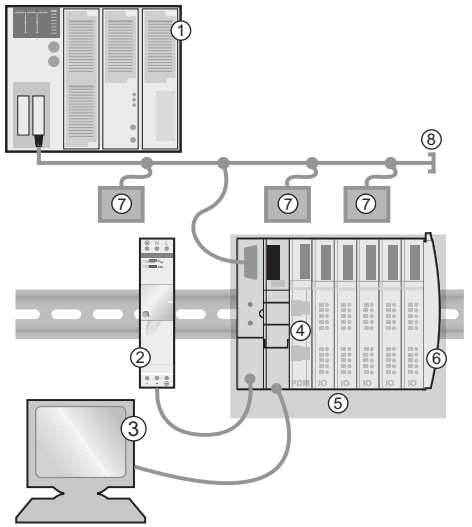
Integrated Power Supply

The NIM's built-in 24-to-5 VDC power supply provides logic power to the I/O modules on the primary segment of the island bus. The power supply requires a 24 VDC external power source. It converts the 24 VDC to 5 V of logic power for the island. Individual STB I/O modules in an island segment generally draw a logic bus current of between 50 and 265 mA. (Consult the *Advantys STB System Planning and Installation Guide* for current limitations at various operating temperatures.) If the logic bus current drawn by the I/O modules totals more than 1.2 A, additional STB power supplies need to be installed to support the load.

The NIM delivers the logic power signal to the primary segment only. Special STB XBE 1300 beginning-of-segment (BOS) modules, located in the first slot of each extension segment, have their own built-in power supplies, which provide logic power to the STB I/O modules in the extension segments. Each BOS module that you install requires 24 VDC from an external power supply.

Structural Overview

The following figure illustrates the multiple roles of the NIM. The figure provides a network view and a physical representation of the island bus:



1 fieldbus master
2 external 24 VDC power supply, the source for logic power on the island
3 external device connected to the CFG port (a computer running the Advantys Configuration Software or an HMI panel)
4 power distribution module (PDM): provides field power to the I/O modules
5 island node
6 island bus terminator plate
7 other nodes on the fieldbus network
8 fieldbus network terminator (if required)

What Is Advantys STB?

Introduction

Advantys STB is an assembly of distributed I/O, power, and other modules that function together as an island node on an open fieldbus network. Advantys STB delivers a highly modular and versatile slice I/O solution for the manufacturing and process industries.

Advantys STB lets you design an island of distributed I/O where the I/O modules can be installed as close as possible to the mechanical field devices that they control. This integrated concept is known as *mechatronics*.

Island Bus I/O

An Advantys STB island can support as many as 32 I/O modules. These modules may be Advantys STB I/O modules, preferred modules, and enhanced CANopen devices.

The Primary Segment

STB I/O modules on an island may be interconnected in groups called segments.

Every island has at least one segment, called the *primary segment*. It is always the first segment on the island bus. The NIM is the first module in the primary segment. The primary segment must contain at least one Advantys STB I/O module and can support a logic bus current of up to 1.2 A. The segment also contains one or more power distribution modules (PDMs), which distribute field power to the I/O modules.

Extension Segments

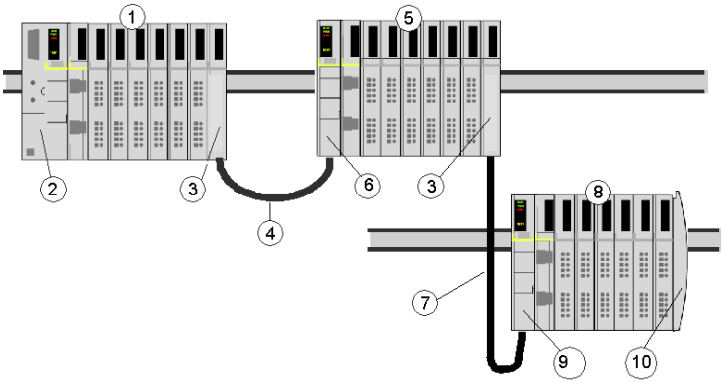
When you are using a standard NIM, Advantys STB I/O modules that do not reside in the primary segment can be installed in *extension segments*. Extension segments are optional segments that enable an island to be a truly distributed I/O system. The island bus can support as many as six extension segments.

Special extension modules and extension cables are used to connect segments in a series. The extension modules are:

- STB XBE 1100 EOS module: the last module in a segment if the island bus is extended
- STB XBE 1300 BOS module: the first module in an extension segment

The BOS module has a built-in 24-to-5 VDC power supply similar to the NIM. The BOS power supply also provides logic power to the STB I/O modules in an extension segment.

Extension modules are connected by lengths of STB XCA 100*x* cable that extend the island communication bus from the previous segment to the next BOS module:



1 primary segment
2 NIM
3 STB XBE 1100 EOS bus extension module(s)
4 1 m length STB XCA 1002 bus extension cable
5 first extension segment
6 STB XBE 1300 BOS bus extension module for the first extension segment
7 4.5 m length STB XCA 1003 bus extension cable
8 second extension segment
9 STB XBE 1300 BOS bus extension module for the second extension segment
10 STB XMP 1100 termination plate

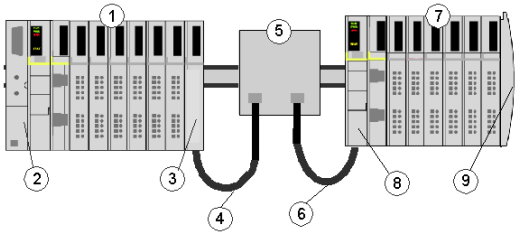
Bus extension cables are available in various lengths, ranging from 0.3 m (1 ft) to 14.0 m (45.9 ft).

Preferred Modules

An island bus can also support those auto-addressable modules referred to as *preferred modules*. Preferred modules do not mount in segments, but they do count as part of the 32-module maximum system limit.

A preferred module can connect to an island bus segment through an STB XBE 1100 EOS module and a length of STB XCA 100*x* bus extension cable. Each preferred module has two IEEE 1394-style cable connectors, one to receive the island bus signals and the other to transmit them to the next module in the series. Preferred modules are also equipped with termination, which must be enabled if a preferred module is the last device on the island bus and must be disabled if other modules follow the preferred device on the island bus.

Preferred modules can be chained to one another in a series, or they can connect to Advantys STB segments. As shown in the following figure, a preferred module passes the island bus communications signal from the primary segment to an extension segment of Advantys STB I/O modules:



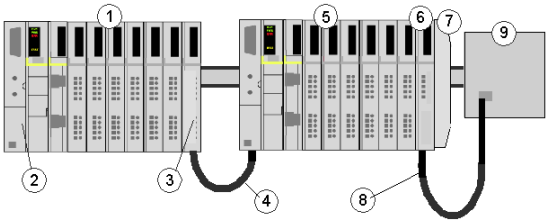
1 primary segment
2 NIM
3 STB XBE 1100 EOS bus extension module
4 1 m length STB XCA 1002 bus extension cable
5 preferred module
6 1 m length STB XCA 1002 bus extension cable
7 extension segment of Advantys STB I/O modules
8 STB XBE 1300 BOS bus extension module for the extension segment
9 STB XMP 1100 termination plate

Enhanced CANopen Devices

You may also install one or more enhanced CANopen devices on an island. These devices are not auto-addressable, and they must be installed at the end of the island bus. If you want to install enhanced CANopen devices on an island, you need to use an STB XBE 2100 CANopen extension module as the last module in the last segment.

NOTE: If you want to include enhanced CANopen devices in your island, you need to configure the island using the Advantys Configuration Software, and you need to configure the island to operate at 500 kbaud.

Because enhanced CANopen devices cannot be auto-addressed on the island bus, they must be addressed using physical addressing mechanisms on the devices. The enhanced CANopen devices together with the CANopen extension module form a sub -network on the island bus that needs to be separately terminated at the beginning and end. A terminator resistor is included in the STB XBE 2100 CANopen extension module for one end of the extension sub-network; the last device on the CANopen extension must also be terminated with a 120 Ω resistor. The rest of the island bus needs to be terminated after the CANopen extension module with an STB XMP 1100 termination plate:



1 primary segment
2 NIM
3 STB XBE 1100 EOS bus extension module
4 1 m length STB XCA 1002 bus extension cable
5 extension segment
6 STB XBE 2100 CANopen extension module
7 STB XMP 1100 termination plate
8 typical CANopen cable
9 enhanced CANopen device with 120 Ω termination

Length of the Island Bus

The maximum length of an island bus (the maximum distance between the NIM and the last device on the island) is 15 m (49.2 ft). This length must take into account the extension cables between segments, extension cables between preferred modules, and the space consumed by the devices themselves.

About the CANopen Fieldbus Protocol

Introduction

CANopen, a digital communications network, is a defined set of instructions for transmitting data and services in an open CAN environment. CANopen is a standard profile for industrial automation systems based on CAL (the CAN application layer). It is especially suited to real-time automation because it is an efficient, low-cost solution for industrial, embedded, and portable applications.

CANopen specifies a communication profile (DS-301) and a set of device profiles (DS-401, DSP-402, etc.).

General system features, like synchronized data exchange, event and error notification, and system-wide timing mechanisms are also defined.

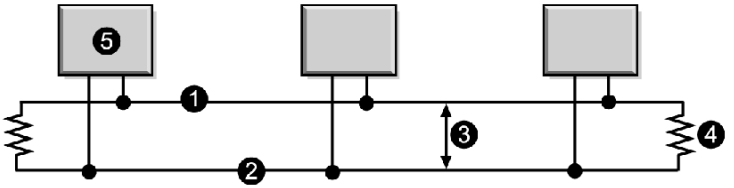
NOTE: For more on standard CANopen specifications and mechanisms, refer to the CiA home page (http//www.can-cia.de/).

Physical Layer

CAN employs a differentially driven (common return), two-wire bus line. A CAN signal is the difference between the voltage levels of the CAN-high and CAN-low wires. (See the figure below).

CAN Bus Line

The figure shows the physical layer components on a two-wire CAN bus:



1 CAN-high wire
2 CAN-low wire
3 difference between the CAN-high/CAN-low voltage signals
4 120 Ω termination
5 node

Bus wires can be routed in parallel or twisted or shielded, depending on EMC requirements. A single line structure minimizes reflection.

EMI

The CAN physical layer is not highly susceptible to EMI because the *difference* in the two wires is unchanged when both wires are affected equally by interference.

Node Limitations

A CANopen network is limited to 128 nodes (node IDs 0 to 127).

Maximum Network Lengths

The following table shows the range of bauds that the STB NCO 2212 CANopen NIM supports for CAN devices and the resulting maximum length of the CANopen network.

Baud	CANopen Network Length
1 mbits/s	25 m
800 kbits/s	50 m
500 kbits/s	100 m
250 kbits/s	250 m
125 kbits/s	500 m
50 kbits/s	1000 m
20 kbits/s	2500 m
10 kbits/s	5000 m

Producer/Consumer Model

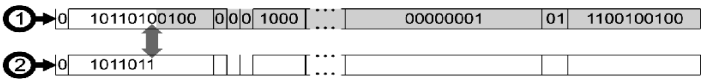
Like any broadcast communications network, CANopen operates within a producer/consumer model. All nodes *listen* on the network for messages that apply to their functionality (according to information in their own object dictionaries). Messages sent by producer devices will be accepted only by particular consumer devices. CANopen also employs the client/server and master/slave models.

Message Prioritization and Arbitration

At any given time, only one node has write access to the CANopen bus. If a node is transmitting on the bus, all other nodes must wait for the bus to be free before attempting a transmission.

CAN data frames have an arbitration field that includes the message identifier field and a remote transmission request bit. When two messages collide while attempting to access the physical layer at the same time, the transmitting nodes perform bitwise arbitration on each other's arbitration fields.

The figure shows the arbitration of the two fields:



1 message with the dominant bit (0)
2 message with the recessive bit (1)

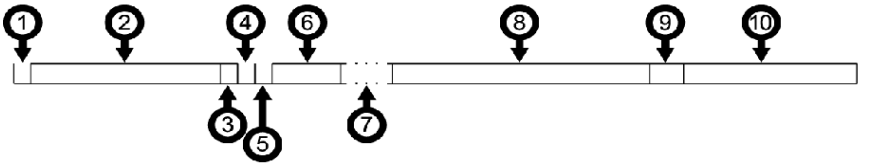
As long as the bits in the arbitration fields have matching values (like the first six bits in the example), they are transmitted on the fieldbus. When the binary values differ (as they do for the seventh bit), the lower value (0) overrides the higher (1). Therefore, message 1 is established as dominant and the transmitting node simply continues to send the remainder of the message data (the shaded area) on the bus.

When the bus is free after the complete transmission of message 1, the transmitting node for message 2 will attempt to access the bus again.

NOTE: Message priority (as a binary value) is determined during system design. Identifiers must be unique to avoid the risk of identical identifiers being associated with different data.

Data Frame Identification

A CANopen data frame can comprise 46 to 110 bits:



1 start (1 bit)
2 identifier (11 bits): low value = high priority (0 = highest priority)
3 remote transmission request (RTR) (1 bit)
4 identifier extension (IDE) (1 bit): first bit of 6-bit control field
5 r0 (1 bit): reserved
6 data length code (DLC) (4 bits): data length for code in field 7
7 data field (0-64 bits [0-8 bytes]): application data of the message
8 cyclic redundancy check (including CRC delimiter) (15 bits) = high (recessive): checksum for preceding message bits
9 ACK field (2 bits) (including ACK delimiter = high (recessive)
10 end of frame (EOF) and inter frame space (IFS) (10 bits)

Object Dictionary

The object dictionary *(see page 68)* is the most important part of the device model *(see page 65)* because it is a map to the internal structure of a particular CANopen device (according to CANopen profile DS-401).

Electronic Data Sheet

The EDS (electronic data sheet *(see page 64)*) is an ASCII file that contains information about a device's communications functionality and the objects in its object dictionary (according to DS-301). Device-specific and manufacturer-specific objects are also defined in the EDS (CiA standards DS-401 and DSP-402).

Each CANopen module's objects and communications functionality are described in its EDS. The EDS specifies the implemented object dictionary entries for a particular device. Only configurable objects are described in the EDS.

The STB NCO 2212 NIM Module

2

Introduction

This chapter describes the STB NCO 2212 standard NIM's external features, connections, power requirements, and product specifications.

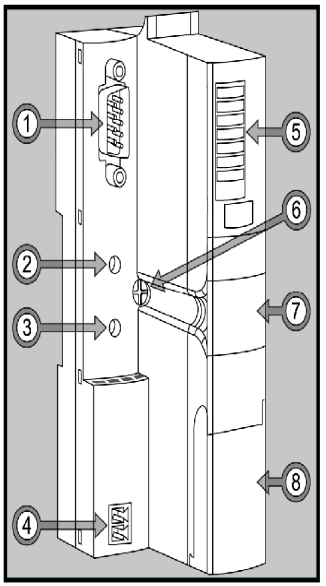
What's in this Chapter?

This chapter contains the following topics:

External Features of the STB NCO 2212 NIM

Introduction

The physical features critical to STB NCO 2212 CANopen NIM operations are called out in the illustration below:



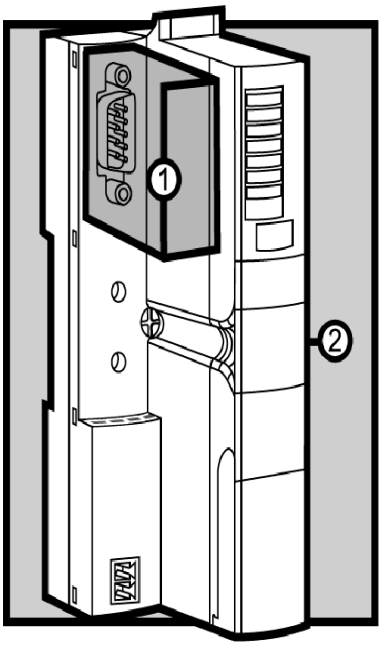
The features in the above illustration are described briefly in the following table:

Feature		Function
1	fieldbus interface *(see page 24)*	A nine-pin SUB-D connector used to connect the NIM and the island bus to a CANopen fieldbus.
2	upper rotary switch	The two rotary switches *(see page 26)* are used together to specify the NIM's node ID on the CANopen fieldbus and to set the fieldbus baud value at the NIM.
3	lower rotary switch	
4	power supply interface *(see page 38)*	A two-receptacle connector for connecting an external 24 VDC power supply to the NIM.
5	LED array *(see page 30)*	Colored LEDs that use various patterns to visually indicate the operational status of the island bus.
6	release screw	A mechanism used for removing the NIM from the DIN rail. (See the *Advantys STB System Planning and Installation Guide* for details.)

Feature		Function
7	removable memory card drawer	A plastic drawer in which a removable memory card *(see page 52)* can be seated and then inserted into the NIM.
8	CFG port cover	A hinged flap on the NIM's front panel that covers the CFG interface *(see page 35)* and the RST button *(see page 58)*.

Housing Shape

The L-shaped external housing of the NIM is designed to accommodate the attachment of a fieldbus connector without raising the depth profile of the island:



1 space reserved for the network connector
2 NIM housing

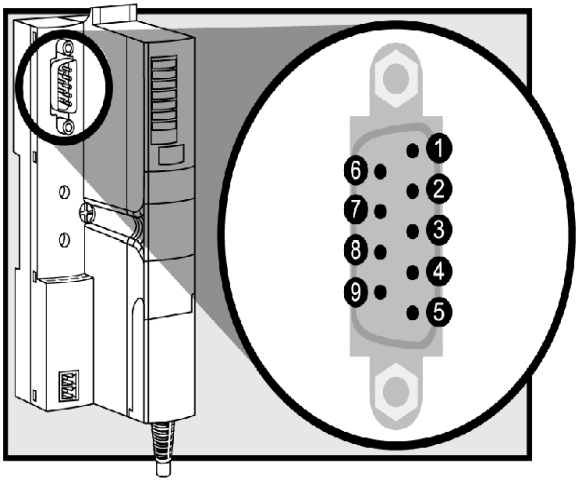
CANopen Fieldbus Interface

Summary

The fieldbus interface on the front of the module is the point of connection between the Advantys STB I/O modules and the CANopen network. The interface is a nine-pin SUB-D (DB-9P) connector.

Fieldbus Port Connections

The fieldbus interface is located on the front of the module at the top:



It is recommended that you use a 9-pin SUB-D (DB-9S) connector compliant with DIN 41652 or corresponding international standard. The pin-out should be according to the table below:

Pin	Signal	Description
1	Unused	Reserved
2	CAN_L	CAN-low bus line
3	CAN_GND	CAN ground
4	Unused	Reserved
5	CAN_SHLD	optional CAN shield
6	GND	optional ground
7	CAN_H	CAN-high bus line
8	Unused	Reserved
9	Unused	Reserved
Note: Pin numbers correspond to callouts in the figure above.		

CANopen Networking Cable and Connectors

The drop cable from the fieldbus to the island must have a DB-9S connector that observes the above pin assignment scheme. The CANopen networking cable is a shielded, twisted-pair electrical cable, compliant with CANopen standard CiA DR-303-1. There should not be an interruption to any wire in the bus cable. This allows for a future specification for use of reserved pins.

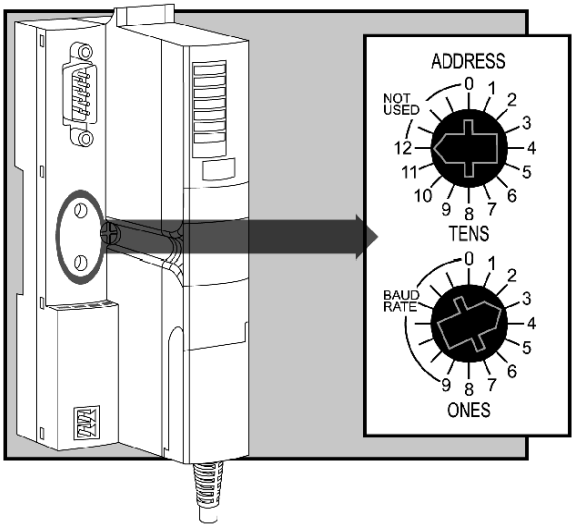
Rotary Switches: Setting the Baud and Network Node Address

Summary

The rotary switches on the STB NCO 2212 CANopen NIM are used to set the Advantys STB island's node address and baud.

Physical Description

The two rotary switches are located on the front of the CANopen NIM, below the fieldbus connection port. Each switch has sixteen positions.



The Baud

The NIM detects a new baud selection in the rotary switches only during power up. The baud is written to nonvolatile Flash memory. It is overwritten only if the NIM detects a change in the baud selection switches during a subsequent power up. In all likelihood, you will rarely change this setting because your system's baud requirements are not likely to change over the short term.

On the lower switch (BAUD RATE), positions 0 through 9 are labeled incrementally on the housing. Setting the lower switch to any of the last six unmarked positions allows you to set a particular baud with the upper switch (ADDRESS).

Setting the Baud

Instructions for setting the baud are in the table.

Step	Action	Comment
1	Bring the power down on the island.	The NIM will detect the changes you are about to make only at the next power up.
2	With a small screwdriver, set the bottom rotary switch to any position after 9 (BAUD RATE).	Setting the switch to any of these unmarked positions prepares the NIM to accept a new baud.
3	Decide on the baud you will employ for fieldbus communications.	The baud setting is according to your system and network requirements.
4	Determine the upper switch position that corresponds to the selected baud.	Use the baud selection table below.
5	With a small screwdriver, set the upper rotary switch to the position that corresponds to your selected baud.	Use the switch position you selected in the last step.
6	Power up your island to employ the new setting.	The NIM reads the rotary switch settings only during power up.

Baud Selection Table

When the lower switch is turned to any one of its baud rate positions, the baud is defined by the position on the upper switch. Only positions 0 through 7 are used to set the baud.

Position (Upper Switch)	Baud
0	10,000 bits/s
1	20,000 bits/s
2	50,000 bits/s
3	125,000 bits/s
4	250,000 bits/s
5	500,000 bits/s
6	800,000 bits/s
7	1 Mbits/s

NOTE: The default baud in Flash memory for a new STB NCO 2212 CANopen NIM is 1 Mbits/s.

The Node Address

Because the CANopen fieldbus master sees the Advantys STB island as *one* network node, the island has a single fieldbus network address. Unlike the baud, the node address is not stored in Flash memory. The NIM reads the node address from the rotary switches each time the island powers up.

The address can be any numeric from 1 to 127 that is unique with respect to other nodes on the network. The fieldbus master and the island bus can communicate over the CANopen network only while the NIM's rotary switches are set to a valid address *(see page 29)*.

Setting the Node Address

Instructions for setting the node address are in the table.

Step	Action	Comment
1	Be sure you have set the desired baud (with the procedure above) *before* setting the node address.	If you set the baud *after* setting the node address, the system will not read a node address from the rotary switches at the next startup.
2	Bring the power down on the island.	The changes you are about to make will be detected only at the next power up.
3	Select a node address that is currently available on your fieldbus network.	Your list of active fieldbus nodes indicates whether a particular address is available.
4	With a small screwdriver, set the lower rotary switch to the position that represents the digit in the ones position of your selected node address.	For example, for a node address of 96, set the lower switch to 6.
5	With a small screwdriver, set the upper rotary switch to the position that represents the two digits in the tens and hundreds position of your selected node address.	For example, for a node address of 96, set the upper switch to 9.
6	Power up Advantys STB.	The NIM reads the rotary switch settings only during power up.

Using the Node Address

After configuring the island's fieldbus network address, it is best to simply leave the rotary switches set to that address. In this way, the CANopen network always identifies the island as the same node address at each power up.

Valid CANopen Node Addresses

Each rotary switch position that you can use to set the node address for your island is marked incrementally on the NIM housing. The available positions on each rotary switch are:

- upper switch—0 to 12 (tens digit)
- lower switch—0 to 9 (ones digit)

For example, the figure *(see page 26)* at the beginning of this topic shows an address of 123 represented by the selection of 3 on the lower switch and 12 on the upper switch.

Note that it is *mechanically* possible to set any node address from 00 to 129, however, addresses 128 and 129 are not available because CANopen supports only 128 node addresses (0 to 127). Also, 00 is never used as a CANopen node address.

Communicating on the Fieldbus

The NIM will only communicate with the fieldbus network while the rotary switches are set to a valid CANopen node address *(see page 29)*. If the bottom switch is in the baud rate position (or if both switches represent an invalid CANopen address), the NIM will wait for you to set a node address before it begins to communicate on the fieldbus. Therefore, configure the desired baud *before* assigning the island's node address to avoid having to reset the address switches later.

If the island has an invalid node address, it cannot communicate with the master. To establish communication, set the switches to a valid address and cycle power on the island.

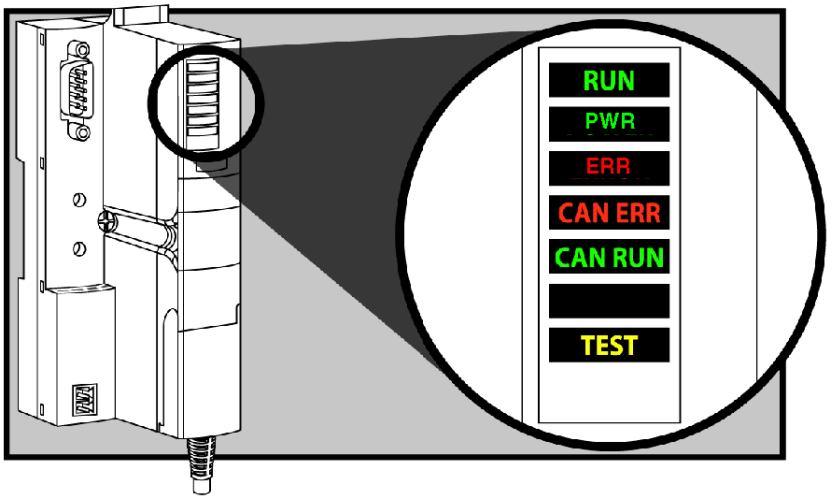
LED Indicators

LED Location

Six LEDs on the STB NCO 2212 NIM visually indicate the operational status of the island bus on a CANopen network. The LED array is located at the top of the NIM front bezel:

- LED 4 (CAN ERR) and LED 5 (CAN RUN) *(see page 31)* indicate the status of data exchange between the CANopen fieldbus master and the Advantys STB island bus.
- LEDs 1, 2, 3, and 7 indicate activity and/or events on the NIM. *(see page 32)*
- LED 6 is not used.

The illustration shows the six LEDs used by the Advantys STB CANopen NIM:



Blink Patterns for CANopen Communications

Individual blinks are approximately 200 ms. There is a 1-second interval between blink sequences. For example:

- blinking: blinks steadily, alternating between 200 ms on and 200 ms off.
- blink 1: blinks once (200 ms), then 1 second off.
- blink 2: blinks twice (200 ms on, 200 ms off, 200 ms on), then 1 second off.
- blink *N*: blinks N (some number of) times, then 1 second off.

NOTE: It is assumed that the *PWR* LED is on continuously, indicating that the NIM is receiving adequate power. *(see page 32)* If the *PWR* LED is off, logic power *(see page 40)* to the NIM is off or insufficient.

CANopen Communications LEDs

The following table describes the indicated condition(s) and the colors and blink patterns that the CAN ERR and CAN RUN LEDs use to show normal operations and error conditions for an Advantys STB CANopen NIM on a CANopen fieldbus.

Label	Pattern	Meaning
CAN ERR (red)	off	No error.
	blinking	Invalid node address on rotary switches.
	on	CAN controller gets reset, Rx/Tx queues cleared, COBs lost.
	blink 1	CAN controller error status bit set; error warning limit reached.
	blink 2	Guardfail or heartbeat failure—node not guarded within lifetime, or heartbeat failure.
	blink *n*	Island bus error. *(see page 32)*
CAN RUN (green)	off	Reset or initialize island bus.
	steady blink	Island bus is pre-operational.
	on	Island bus is operational.
	blink 1	Island bus is stopped.

Advantys STB Island Status LEDs

About the Island Status LEDs

The following table describes:

- the island bus condition(s) communicated by the LEDs
- the colors and blink patterns used to indicate each condition

As you refer to the table, keep in mind the following:

- It is assumed that the *PWR* LED is on continuously, indicating that the NIM is receiving adequate power. If the *PWR* LED is off, logic power *(see page 40)* to the NIM is off or insufficient.
- Individual blinks are approximately 200 ms. There is a 1-second interval between blink sequences. Please note:
  - blinking: blinks steadily, alternating between 200 ms on and 200 ms off.
  - blink 1: blinks once (200 ms), then 1 second off.
  - blink 2: blinks twice (200 ms on, 200 ms off, 200 ms on), then 1 second off.
  - blink *N*: blinks N (some number of) times, then 1 second off.
  - If the *TEST* LED is on, either the Advantys configuration software or an HMI panel is the master of the island bus. If the *TEST* LED is off, the fieldbus master has control of the island bus.

Island Status LED Indicators

RUN (green)	ERR (red)	TEST (yellow)	Meaning
blink: 2	blink: 2	blink: 2	The island is powering up (self test in progress).
off	off	off	The island is initializing. The island is not started.
blink: 1	off	off	The island has been put in the pre-operational state by the RST button. The island is not started.
		blink: 3	The NIM is reading from the removable memory card *(see page 55)*.
		on	The NIM is overwriting its Flash memory with the card's configuration data. (See note 1.)
off	blink: 8	off	The contents of the removable memory card are invalid.
blinking (steady)	off	off	The NIM is configuring *(see page 47)* or auto-configuring *(see page 51)* the island bus. The island bus is not started.
blinking	off	on	Auto-configuration data is being written to Flash memory. (See note 1.)
blink: 3	blink: 2	off	Configuration mismatch detected after power up. At least one mandatory module does not match. The island bus is not started.

RUN (green)	ERR (red)	TEST (yellow)	Meaning
off	blink: 2	off	The NIM has detected a module assignment error; the island bus is not started.
	blink: 5		invalid internal triggering protocol
off	blink: 6	off	The NIM detects no I/O modules on the island bus.
	blinking (steady)	off	The NIM detects no I/O modules on the island bus ... or ...
			No further communications with the NIM are possible. Probable causes: ● internal condition ● wrong module ID ● device did not auto-address *(see page 48)* ● mandatory module is incorrectly configured *(see page 130)* ● process image is not valid ● device is incorrectly configured *(see page 51)* ● The NIM has detected an anomaly on the island bus. ● receive/transmit queue software overrun
on	off	off	The island bus is operational.
on	blink 3	off	At least one standard module does not match. The island bus is operational with a configuration mismatch.
on	blink: 2	off	There is a serious configuration mismatch (when a module is pulled from a running island). The island bus is now in pre-operational mode because of one or more mismatched mandatory modules.
blink: 4	off	off	The island bus is stopped (when a module is pulled from a running island). No further communications with the island are possible.
off	on	off	Internal condition: The NIM is inoperable.
[any]	[any]	on	Test mode is enabled: The configuration software or an HMI panel can set outputs. (See note 2.)

1 The TEST LED is on temporarily during the Flash overwrite process.

2 The TEST LED is on steadily while the device connected to the CFG port is in control.

Power LED

The PWR (power) LED indicates whether or not the STB NIC 2212's internal power supplies are operating at the correct voltages. The PWR LED is directly driven by the STB NIC 2212's reset circuitry.

The following table summarizes the PWR LED states:

Label	Pattern	Meaning
PWR	Steady on	The STB NIC 2212 internal voltages are all at or above their minimum voltage level.
PWR	Steady off	One or more of the STB NIC 2212 internal voltages is below minimum voltage level.

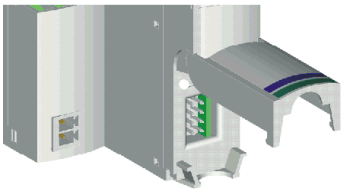
The CFG Interface

Purpose

The CFG port is the connection point to the island bus for either a computer running the Advantys Configuration Software or an HMI panel.

Physical Description

The CFG interface is a front-accessible RS-232 interface located behind a hinged flap on the bottom front of the NIM:



The port uses an 8-pin HE-13 (male) connector.

Port Parameters

The CFG port supports the set of communication parameters listed in the following table. If you want to apply any settings other than the factory default values, you must use the Advantys Configuration Software:

Parameter	Valid Values	Factory Default Settings
bit rate (baud)	2400/4800/9600/19200/ 38400/ 57600	9600
data bits	7/8	8
stop bits	1 or 2	1
parity	none / odd / even	even
Modbus communications mode	RTU	RTU

NOTE: To restore all of the CFG port's communication parameters to their factory default settings, push the RST button *(see page 58)* on the NIM. Be aware, however, that this action overwrites all of the island's current configuration values with factory default values.

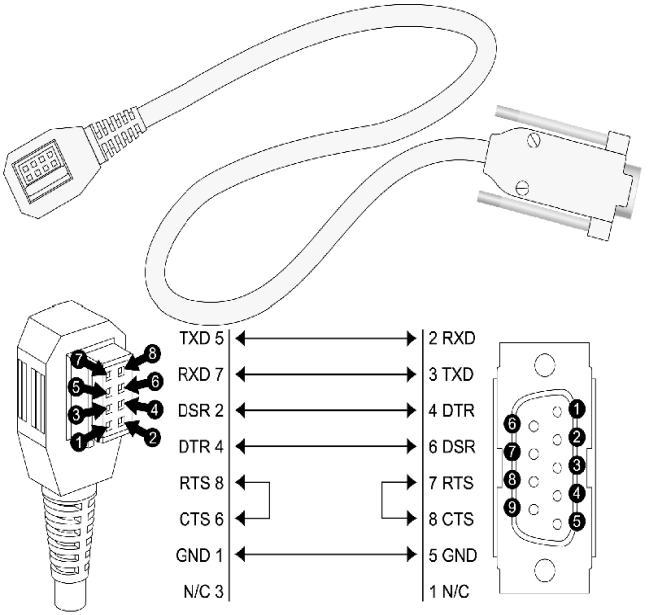
If you want to retain your configuration and still use the RST button to reset your port parameters, write the configuration to an STB XMP 4440 removable memory card *(see page 52)* and insert the card in its drawer in the NIM.

You can also password-protect a configuration *(see page 140)*. If you do this, however, the RST button is disabled and you are unable to use it to reset the port parameters.

Connections

An STB XCA 4002 programming cable must be used to connect the computer running the Advantys Configuration Software or a Modbus-capable HMI panel to the NIM through the CFG port.

The STB XCA 4002 is a 2 m (6.23 ft) shielded, twisted-pair cable with an 8-receptacle HE-13 (female) connector on one end that plugs into the CFG port and a 9-receptacle SUB-D (female) connector on the other end that plugs into a computer or an HMI panel:



TXD transmit data
RXD receive data
DSR data set ready
DTR data terminal ready
RTS request to send
CTS clear to send
GND ground reference
N/C not connected

The following table describes the specifications for the programming cable:

Parameter	Description
model	STB XCA 4002
function	connection to a device running the Advantys Configuration Software
	connection to an HMI panel
communications protocol	Modbus, either RTU or ASCII mode
cable length	2 m (6.23 ft)
cable connectors	• 8-receptacle HE-13 (female) • 9-receptacle SUB-D (female)
cable type	multiconductor

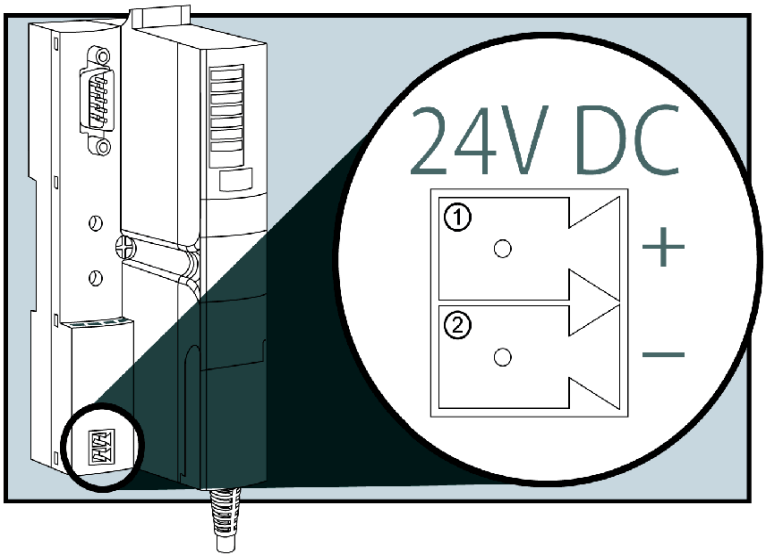
Power Supply Interface

Introduction

The NIM's built-in power supply requires 24 VDC from an external SELV-rated power source. The connection between the 24 VDC source and the Advantys STB island is the two-receptacle connector illustrated below.

Physical Description

Power from the external 24 VDC supply comes in to the NIM through a two-receptacle connector located at the bottom left of the module:

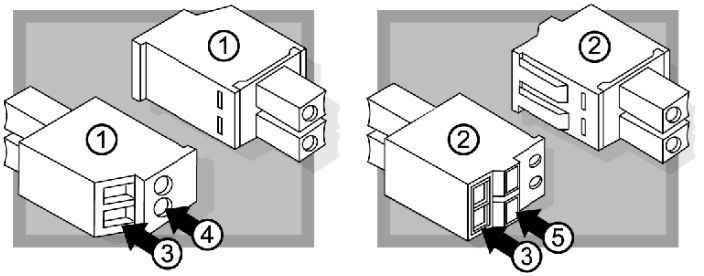


1 receptacle 1—24 VDC
2 receptacle 2—common

Connectors

Screw-type and spring-type connectors are provided with the NIM. Replacement connectors are also available.

The following illustrations show two views of each power connector type. A front and back view of the STB XTS 1120 screw type connector is shown on the left, and a front and back view of the STB XTS 2120 spring clamp connector is shown on the right:



1 STB XTS 1120 screw-type power connector
2 STB XTS 2120 spring clamp power connector
3 wire entry slot
4 screw clamp access
5 spring clamp actuation button

Each entry slot accepts a wire in the range 0.14 to 1.5 mm^2 (28 to 16 AWG).

Logic Power

Introduction

Logic power is a 5 VDC power signal on the island bus that the I/O modules require for internal processing. The NIM has a built-in power supply that provides logic power. The NIM sends the 5 V logic power signal across the island bus to support the modules in the primary segment.

External Source Power

⚠ CAUTION

IMPROPER GALVANIC ISOLATION

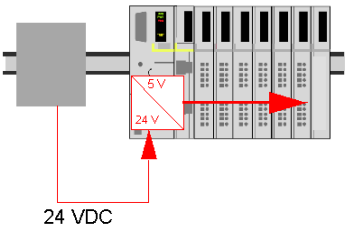
The power components are not galvanically isolated. They are intended for use only in systems designed to provide SELV isolation between the supply inputs or outputs and the load devices or system power bus. You must use SELV-rated supplies to provide 24 VDC source power to the NIM.

Failure to follow these instructions can result in injury or equipment damage.

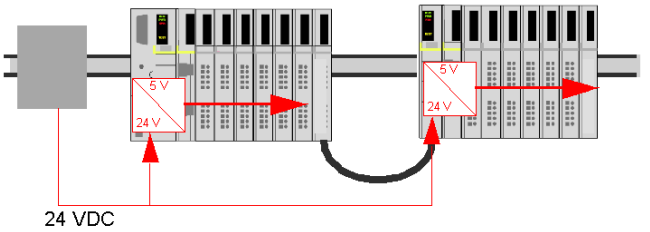
Input from an external 24 VDC power supply *(see page 42)* is needed as the source power for the NIM's built-in power supply. The NIM's built-in power supply converts the incoming 24 V to 5 V of logic power. The external supply must be rated *safety extra low voltage* (SELV-rated).

Logic Power Flow

The figure below shows how the NIM's integrated power supply generates logic power and sends it across the primary segment:



The figure below shows how the 24 VDC signal is distributed to an extension segment across the island:



The logic power signal is terminated in the STB XBE 1100 module at the end of the segment (EOS).

Island Bus Loads

The built-in power supply provides logic bus current to the island. If the logic bus current drawn by the I/O modules exceeds the available current, install additional STB power supplies to support the load. Consult the *Advantys STB System Planning and Installation Guide* for the current provided and consumed by Advantys STB modules at various operating temperatures and voltages.

Selecting a Source Power Supply for the Island's Logic Power Bus

Logic Power Requirements

An external 24 VDC power supply is needed as the source for logic power to the island bus. The external power supply connects to the island's NIM. This external supply provides the 24 V input to the built-in 5 V power supply in the NIM.

The NIM delivers the logic power signal to the primary segment only. Special STB XBE 1300 beginning-of-segment (BOS) modules, located in the first slot of each extension segment, have their own built-in power supplies, which provide logic power to the STB I/O modules in the extension segments. Each BOS module that you install requires 24 VDC from an external power supply.

Characteristics of the External Power Supply

⚠ CAUTION

IMPROPER GALVANIC ISOLATION

The power components are not galvanically isolated. They are intended for use only in systems designed to provide SELV isolation between the supply inputs or outputs and the load devices or system power bus. You must use SELV-rated supplies to provide 24 VDC source power to the NIM.

Failure to follow these instructions can result in injury or equipment damage.

The external power supply needs to deliver 24 VDC source power to the island. The supply that you select can have a low range limit of 19.2 VDC and a high range limit of 30 VDC. The external supply must be rated *safety extra low voltage* (SELV-rated).

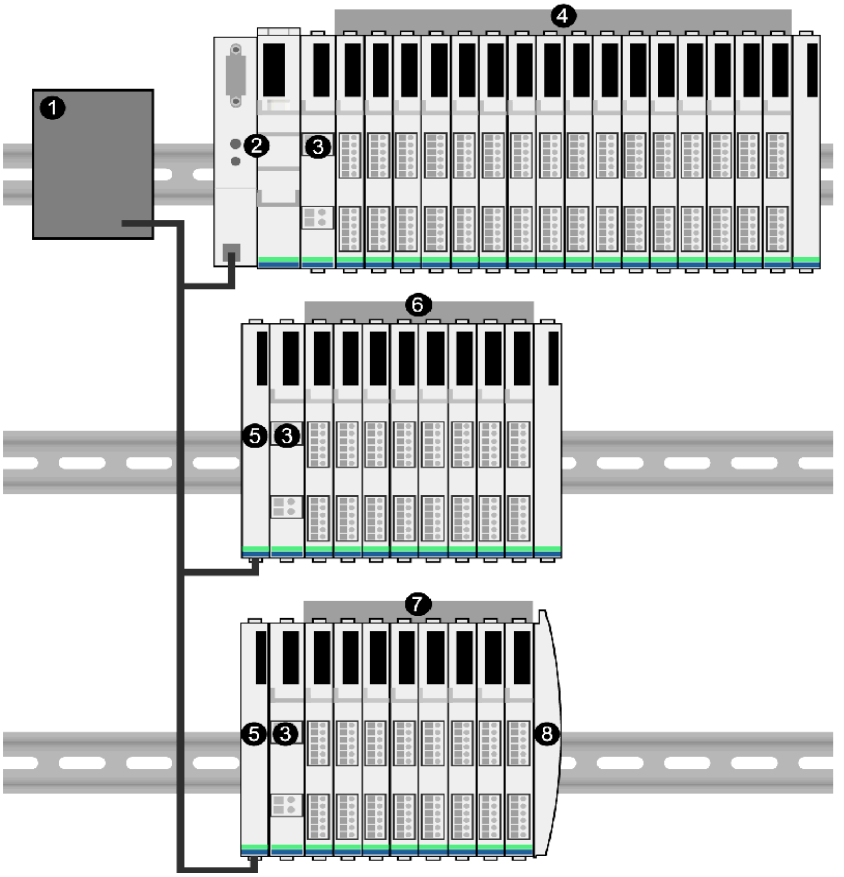
The SELV rating means that, in addition to basic insulation between hazardous voltages and the DC output, a second supplementary insulation layer has been added. As a result, if a single component/insulation does not perform, the DC output does not exceed SELV limits.

Calculating the Wattage Requirement

The amount of power *(see page 40)* that the external power supply must deliver is a function of the number of modules and the number of built-in power supplies installed on the island.

The external supply needs to provide 13 W of power for the NIM and 13 W for each additional STB power supply (like an STB XBE 1300 BOS module). For example, a system with one NIM in the primary segment and one BOS module in an extension segment would require 26 W of power.

Here is an example of an extended island:



1 24 VDC source power supply
2 NIM
3 PDM
4 primary segment I/O modules
5 BOS module

6 first extension segment I/O modules
7 second extension segment I/O modules
8 island bus terminator plate

The extended island bus contains three built-in power supplies:

- the supply built into the NIM, which resides in the leftmost location of the primary segment
- a power supply built into each of the STB XBE 1300 BOS extension modules, which reside in the leftmost location of the two extension segments

In the figure, the external supply would provide 13 W of power for the NIM plus 13 W for each of the two BOS modules in the extension segments (for a total of 39 W).

NOTE: If the 24 VDC source power supply also supplies field voltage to a power distribution module (PDM), you must add the field load to your wattage calculation. For 24 VDC loads, the calculation is simply *amps* x *volts* = *watts*.

Suggested Devices

The external power supply is generally enclosed in the same cabinet as the island. Usually the external power supply is a DIN rail-mountable unit.

We recommend using ABL8 Phaseo power supplies.

Module Specifications

Overview

This information describes general specifications for the NIM.

Specifications Detail

The following table lists the system specifications for the STB NCO 2212 CANopen NIM:

General Specifications		
dimensions	width	40.5 mm (1.59 in)
	height	130 mm (5.12 in)
	depth	70 mm (3.15 in)
interface connectors	to the CANopen network	nine-pin SUB-D connector
	RS-232 port for configuration software or HMI panel	eight-receptacle HE-13
	to the external 24 VDC power supply	two-receptacle
built-in power supply	input voltage	24 VDC nominal
	input power range	19.2 ... 30 VDC
	input current	400 mA @ 24 VDC
	output voltage to the island bus	5 VDC @ 1.2 A
	output current rating	1.2 A @ 5 VDC
	isolation	no internal isolation (isolation must be provided by a SELV-rated external 24 VDC source power supply)
	noise immunity (EMC)	EN 61131-2
addressable I/O modules supported		32 maximum/island
segments supported	primary (required)	one
	extension (optional)	six maximum
standards	CANopen conformance	CiA DS-301
	MTBF	200,000 hours GB (ground benign)
storage temperature		-40 to 85°C
operating temperature range*		0 to 60°C
agency certifications		refer to the *Advantys STB System Planning and Installation Guide, 890 USE 171 00*
*This product supports operation at normal and extended temperature ranges. Refer to the *Advantys STB System Planning and Installation Guide, 890 USE 171 00* for a complete summary of cabalities and limitations.		

How to Configure the Island

3

Introduction

The information in this chapter describes the auto-addressing and auto-configuration processes. An Advantys STB system has an auto-configuration capability in which the actual configuration of I/O modules on the island is read and saved to Flash.

The removable memory card is discussed in this chapter. The card is an Advantys STB option for storing configuration data offline. Factory default settings can be restored to the island bus I/O modules and the CFG port by engaging the RST button.

The NIM is the physical and logical location of all island bus configuration data and functionality.

What's in this Chapter?

This chapter contains the following topics:

How Do Modules Automatically Get Island Bus Addresses?

Introduction

Each time that the island is powered up or reset, the NIM automatically assigns a unique island bus address to each module on the island that engages in data exchange. All Advantys STB I/O modules and preferred devices engage in data exchange and require island bus addresses.

About the Island Bus Address

An island bus address is a unique integer value in the range 1 through 127 that identifies the physical location of each addressable module on the island. The NIM's address is always 127. Addresses 1 through 32 are available for I/O modules and other island devices.

During initialization, the NIM detects the order in which modules are installed and addresses them sequentially from left to right, starting with the first addressable module after the NIM. No user action is required to address these modules.

Addressable Modules

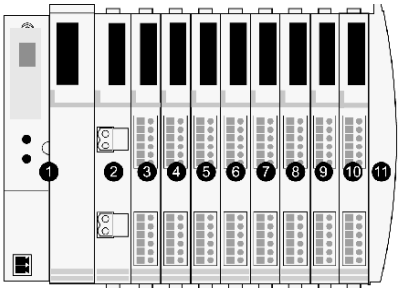
Advantys STB I/O modules and preferred devices are auto-addressable. Enhanced CANopen modules are not auto-addressable. They require manual address settings.

Because they do not exchange data on the island bus, the following are not addressed:

- bus extension modules
- PDMs such as the STB PDT 3100 and STB PDT 2100
- auxiliary power supplies, such as the STB CPS 2111
- termination plate

An Example

For example, if you have an island bus with eight I/O modules:



1 NIM
2 STB PDT 3100 (24 VDC power distribution module)
3 STB DDI 3230 24 VDC (2-channel digital input module)
4 STB DDO 3200 24 VDC (2-channel digital output module)
5 STB DDI 3420 24 VDC (4-channel digital input module)
6 STB DDO 3410 24 VDC (4-channel digital output module)
7 STB DDI 3610 24 VDC (6-channel digital input module)
8 STB DDO 3600 24 VDC (6-channel digital output module)
9 STB AVI 1270 +/-10 VDC (2-channel analog input module)
10 STB AVO 1250 +/-10 VDC (2-channel analog output module)
11 STB XMP 1100 (island bus termination plate)

The NIM would auto-address it as follows. Note that the PDM and the termination plate do not consume island bus addresses:

Module	Physical Location	Island Bus Address
NIM	1	127
STB PDT 3100 PDM	2	not addressed: does not exchange data
STB DDI 3230 input	3	1
STB DDO 3200 output	4	2
STB DDI 3420 input	5	3
STB DDO 3410 output	6	4
STB DDI 3610 input	7	5
STB DDO 3600 output	8	6
STB AVI 1270 input	9	7
STB AVO 1250 output	10	8
STB XMP 1100 termination plate	11	not applicable

Associating the Module Type with the Island Bus Location

As a result of the configuration process, the NIM automatically identifies physical locations on the island bus with specific I/O module types. This feature enables you to hot swap a non-operational module with a new module of the same type.

How to Auto-Configure Default Parameters for Island Modules

Introduction

All Advantys STB I/O modules are shipped with a set of predefined parameters that allow an island to be operational as soon as it is initialized. This ability of island modules to operate with default parameters is known as auto-configuration. Once an island bus has been installed, assembled, and successfully parameterized and configured for your fieldbus network, you can begin using it as a node on that network.

NOTE: A valid island configuration does not require the intervention of the optional Advantys Configuration Software.

About Auto-Configuration

Auto-configuration occurs under these circumstances:

- The island is powered up with a factory default NIM configuration. (If this NIM is subsequently used to create a new island, auto-configuration does not occur when that new island is powered.)
- You push the RST button *(see page 58)*.
- You force an auto-configuration using the Advantys Configuration Software.

As part of the auto-configuration process, the NIM checks each module and confirms that it has been properly connected to the island bus. The NIM stores the default operating parameters for each module in Flash memory.

Customizing a Configuration

In a custom configuration, you can:

- customize the operating parameters of I/O modules
- create reflex actions *(see page 133)*
- add enhanced CANopen standard devices to the island bus
- customize other island capabilities
- configure communication parameters (STB NIP 2311 only)

How to Install the STB XMP 4440 Optional Removable Memory Card

Introduction

⚠ CAUTION

LOSS OF CONFIGURATION: MEMORY CARD DAMAGE OR CONTAMINATION

The card's performance can be degraded by dirt or grease on its circuitry. Contamination or damage may create an invalid configuration.

- Use care when handling the card.
- Inspect for contamination, physical damage, and scratches before installing the card in the NIM drawer.
- If the card does get dirty, clean it with a soft dry cloth.

Failure to follow these instructions can result in injury or equipment damage.

The STB XMP 4440 removable memory card is a 32-kbyte subscriber identification module (SIM) that lets you store *(see page 139)*, distribute, and reuse custom island bus configurations. If the island is in edit mode and a removable memory card containing a valid island bus configuration is inserted in the NIM, the configuration data on the card overwrites the configuration data in Flash memory, and is adopted when the island starts up. When the island is in protected mode, it ignores the presence of a removable memory card.

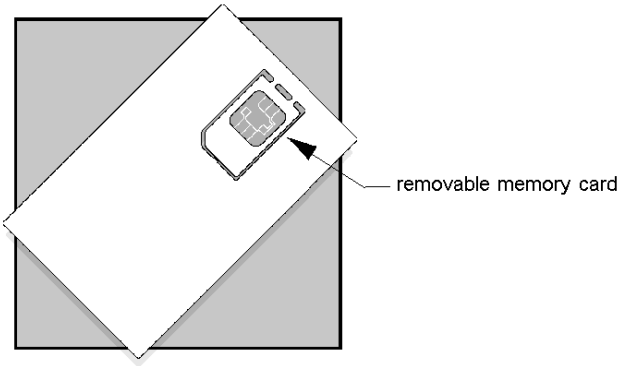
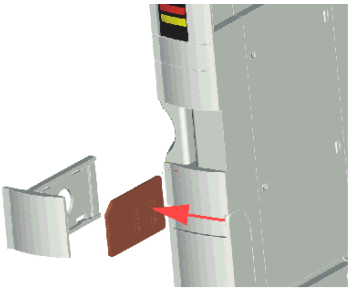
The removable memory card is an optional Advantys STB feature.

Remember:

- Keep the card free of contaminants and dirt.
- Network configuration data, such as the fieldbus baud setting, cannot be saved to the card.

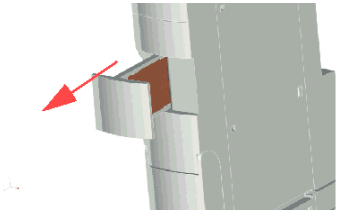
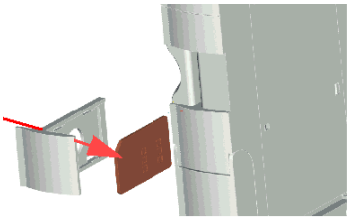
Installing the Card

Use the following procedure to install the memory card:

Step	Action
1	Punch out the removable memory card from the plastic card on which it is shipped.  Make sure that the edges of the card are smooth after you punch it out.
2	Open the card drawer on the front of the NIM. If it makes it easier for you to work, you may pull the drawer completely out from the NIM housing.
3	Align the chamfered edge (the 45° corner) of the removable memory card with the one in the mounting slot in the card drawer. Hold the card so that the chamfer is in the upper left corner. 
4	Seat the card in the mounting slot, applying slight pressure to the card until it snaps into place. The back edge of the card must be flush with the back of the drawer.
5	Close the drawer.

Removing the Card

Use the following procedure to remove the memory card from the NIM. As a handling precaution, avoid touching the circuitry on the card.

Step	Action
1	Open the card drawer. 
2	Push the removable memory card out of the drawer through the round opening at the back. Use a soft but firm object like a pencil eraser. 

Using the STB XMP 4440 Optional Removable Memory Card to Configure the Island

Introduction

A removable memory card is read when an island is powered on or during a reset operation. If the configuration data on the card is valid, the current configuration data in Flash memory is overwritten.

A removable memory card can be *active* only if an island is in *edit* mode. If an island is in protected mode *(see page 140)*, the card and its data are ignored.

Configuration Scenarios

The following discussion describes several island configuration scenarios that use the removable memory card. (The scenarios assume that a removable memory card is already installed in the NIM.):

- initial island bus configuration
- replace the current configuration data in Flash memory in order to:
  - apply custom configuration data to your island
  - temporarily implement an alternative configuration; for example, to replace an island configuration used daily with one used to fulfill a special order
- copying configuration data from one NIM to another, including from a non-operational NIM to its replacement; the NIMs must have the same part number
- configuring multiple islands with the same configuration data

NOTE: Whereas writing configuration data *from* the removable memory card to the NIM does not require use of the optional Advantys Configuration Software, you must use this software to save (write) configuration data *to* the removable memory card in the first place.

Edit Mode

Your island bus must be in edit mode to be configured. In edit mode, the island bus can be written to as well as monitored.

Edit mode is the default operational mode for the Advantys STB island:

- A new island is in edit mode.
- Edit mode is the default mode for a configuration downloaded from the Advantys Configuration Software to the configuration memory area in the NIM.

Initial Configuration and Reconfiguration Scenarios

Use the following procedure to set up an island bus with configuration data that was previously saved *(see page 139)* to a removable memory card. You can use this procedure to configure a new island or to overwrite an existing configuration. (**NOTE:** The use of this procedure destroys your existing configuration data.)

Step	Action	Result
1	Install the removable memory card in its drawer in the NIM *(see page 52)*.	
2	Power on the new island bus.	The configuration data on the card is checked. If the data is valid, it is written to Flash memory. The system restarts automatically, and the island is configured with this data. If the configuration data is invalid, it is not used and the island bus stops. If the configuration data was in edit mode, the island bus remains in edit mode. If the configuration data on the card was password-protected *(see page 140)*, your island bus enters protected mode at the end of the configuration process. **NOTE:** If you are using this procedure to reconfigure an island bus and your island is in protected mode, you can use the configuration software to change the island's operational mode to edit.

Using the Card and the RST Function to Reconfigure an Island

You can use a removable memory card in combination with the RST function to overwrite the island's current configuration data. The configuration data on the card can contain custom configuration features. Using the data on the card, you can add password protection to your island, change the I/O module assembly, and change the user-modifiable CFG port settings *(see page 35)*. *Using this procedure destroys your existing configuration data.*

Step	Action	Comment
1	Place the island bus in edit mode.	If your island is in protected mode, you can use the configuration software to change the island's operational mode to *edit*.
2	Press the **RST** button for at least two seconds.	If your configuration data was in edit mode, the island bus remains in edit mode. If the configuration data on the card was protected, your island bus enters protected mode at the end of the configuration process.

Configuring Multiple Island Buses with the Same Data

You can use a removable memory card to make a copy of your configuration data; then use the card to configure multiple island buses. This capability is particularly advantageous in a distributed manufacturing environment or for an OEM (original equipment manufacturer).

NOTE: The island buses may be either new or previously configured, but the NIMs must all have the same part number.

What is the RST Button?

Summary

The RST function is basically a Flash memory overwriting operation. This means that RST is functional only after the island has been successfully configured at least once. All RST functionality is performed with the RST button, which is enabled only in edit mode *(see page 55)*.

Physical Description

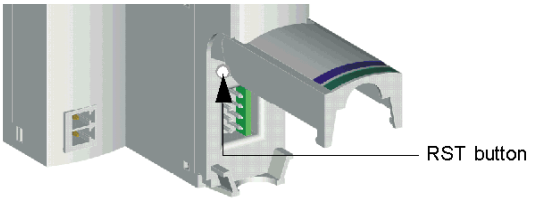
⚠ CAUTION

UNINTENDED EQUIPMENT OPERATION/CONFIGURATION OVERWRITTEN—RST BUTTON

Do not attempt to restart the island with the RST button. Pushing the RST button reconfigures the island with default settings (no custom parameters).

Failure to follow these instructions can result in injury or equipment damage.

The RST button is located immediately above the CFG port *(see page 35)*, and behind the same hinged cover:



Holding down the RST button for 2 seconds or longer causes Flash memory to be overwritten, resulting in a new configuration for the island.

If the island is already auto-configured, there is no consequence other than the island stops during the configuration process. However, island parameters that you previously customized with the Advantys Configuration Software are overwritten by default parameters during the configuration process.

Engaging the RST Button

To engage the RST button, it is recommended that you use a small screwdriver with a flat blade no wider than 2.5 mm (.10 in). Do not use a sharp object that might damage the RST button, nor a soft item like a pencil that might break off and jam the button.

How to Overwrite Flash Memory with the RST Button

Introduction

⚠ CAUTION

UNINTENDED EQUIPMENT OPERATION/CONFIGURATION DATA OVER-WRITTEN—RST BUTTON

Do not attempt to restart the island by pushing the RST button *(see page 58)* causes the island bus to reconfigure itself with factory default operating parameters.

Failure to follow these instructions can result in injury or equipment damage.

The RST function allows you to reconfigure the operating parameters and values of an island by overwriting the current configuration in Flash memory. RST functionality affects the configuration values associated with the I/O modules on the island, the operational mode of the island, and the CFG port parameters.

The RST function is performed by holding down the RST button *(see page 58)* for at least two seconds. The RST button is enabled only in edit mode. In protected mode *(see page 140)*, the RST button is disabled; pressing it has no effect.

NOTE: Pressing the RST button does not affect network settings.

RST Configuration Scenarios

The following scenarios describe some of the ways that you can use the RST function to configure your island:

- Restore factory-default parameters and values to an island, including to the I/O modules and the CFG port *(see page 35)*.
- Add a new I/O module to a previously auto-configured *(see page 51)* island. If a new I/O module is added to the island, pressing the RST button forces the auto-configuration process. The updated island configuration data is automatically written to Flash memory.

Overwriting Flash Memory with Factory Default Values

The following procedure describes how to use the RST function to write default configuration data to Flash memory. Follow this procedure if you want to restore default settings to an island. This is also the procedure to use to update the configuration data in Flash memory after you add an I/O module to a previously auto-configured island bus. *Because this procedure overwrites the configuration data, you may want to save your existing island configuration data to a removable memory card before pushing the RST button.*

Step	Action
1	If you have a removable memory card installed, remove it *(see page 54)*.
2	Place the island in edit mode *(see page 55)*.
3	Hold the RST button *(see page 58)* down for at least two seconds.

The Role of the NIM in this Process

The NIM reconfigures the island bus with default parameters as follows:

Stage	Description
1	The NIM auto-addresses *(see page 48)* the I/O modules on the island and derives their factory-default configuration values.
2	The NIM overwrites the current configuration in Flash memory with configuration data that uses the factory-default values for the I/O modules.
3	It resets the communication parameters on its CFG port to their factory-default values *(see page 35)*.
4	It re-initializes the island bus and brings it into operational mode.

Fieldbus Communications Support

4

Introduction

This chapter describes how the CANopen master sets up communications between itself and an Advantys STB island bus. The chapter describes the parameterization, configuration, and diagnostics services that are performed in order to configure the island bus as a node on a CANopen network.

To communicate with an Advantys STB island, the CANopen master sends output data across its network to the STB NCO 2212 CANopen NIM. The NIM transfers this output data from the master across the island bus to the destination output modules. The NIM will collect input data from the island bus I/O modules. That data is transmitted in bit-packed format over the CANopen network to the fieldbus master.

What's in this Chapter?

This chapter contains the following topics:

The Advantys STB Electronic Data Sheet (EDS)

Introduction

As with any CANopen network node, your Advantys STB island needs to export an electronic data sheet (EDS) to the fieldbus master. The NIM's EDS describes the island configuration as a single node on the CANopen network. By exporting its EDS file to the CANopen master, a node reveals its object dictionary entries to the controlling device.

What's an EDS?

The EDS is a standardized ASCII file that contains information about a network device's communications functionality and the contents of its object dictionary (as defined in DS-301). The EDS also defines device-specific and manufacturer-specific objects (according to DS-401 and DSP-402).

Using the EDS, you can standardize tools to:

- configure CANopen devices
- design networks for CANopen devices
- manage project information on different platforms

The parameters of a particular island configuration depend on those objects (application, communications, parameter, emergency, and other objects) that reside on the individual island modules.

Basic and Configured EDS Files

An EDS that describes the island's basic functionality and objects is included with the STB NCO 2212 CANopen NIM product. Using the basic EDS, you will need to define PDOs *(see page 117)* to access those objects defined within it.

If you wish, you can generate a configuration-specific EDS for your particular island using the (optional) Advantys configuration software.

The Device Model and Communication Objects

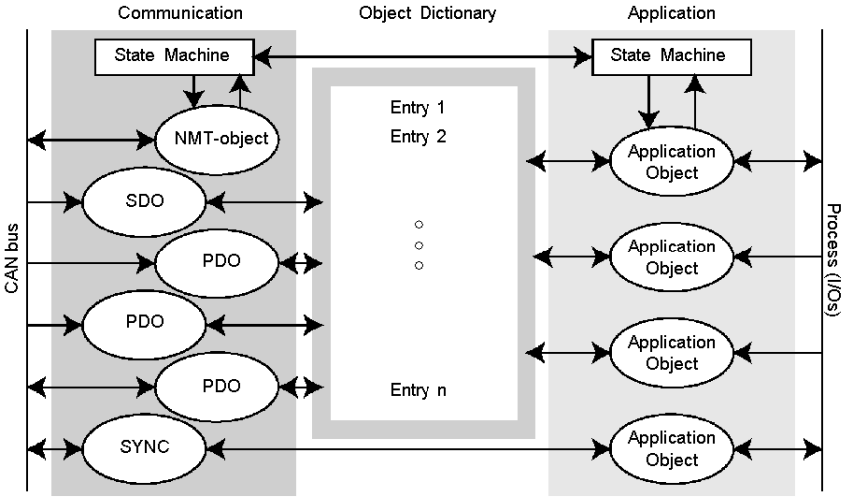
Summary

The interchangeability and interoperability of standard devices in a CANopen system require that the functionality of each device be described to the network in a specific device *profile* that is based on the CANopen device *model*.

Different manufacturers have agreed to standard device profiles for separating industrial automation devices into classes, like encoders, drives, and generic I/O.

The Device Model

The CANopen specification is composed of a set of device profiles that are developed using the device model:



Device Model Components

In CANopen's object-oriented approach, there are basically two types of objects:

- *communication objects*—A communication object (COB) is a unit of transportation (a "message") in a CAN-based network. Data must be sent across a CAN network inside a COB. A COB can contain at most 8 bytes of data. CANopen COBs indicate a particular functionality in a device and are specified in the CANopen communication profile.
- *application objects*—Application objects represent device-specific functionality, such as the state of input or output data. Application objects are specified in the device profile (DS-301).

Advantys STB NIM-Supported Objects

Device objects are accessed through the object dictionary in which they reside. The Advantys STB CANopen NIM supports these objects:

- 32 TxPDOs
- 32 RxPDOs
- 512 device-specific objects
- 512 manufacturer-specific objects
- node guarding
- NMT objects
- 256 transmit objects
- The bytes an SDO can obtain (limited to 20)
- Limitations if the default mapping is used: 1 RxPDO for digital out data (8 bytes); 3 RxPDOs for analog out data (24 bytes); 1 TxPDO for digital in data (8 bytes); 3 TxPDOs for analog in data (24 bytes)

Every CANopen device has a CANopen object dictionary in which parameters for all associated CANopen objects are entered.

Communication Objects

The tables below show the communications objects that CANopen supports. The COB-IDs (communications object identifiers) in the third column are used according to the predefined I/O connection set (DS-301).

This table describes the supported *broadcast* communications objects.

Broadcast Object	Function Code (Binary)	Resulting COB-ID	Communication Parameters at Index
NMT	0000	0	-
SYNC *(see page 96)*	0001	128 (80h)	1005h, 1006h, 1007h

This table describes the supported peer-to-peer COBs.

Peer-to-Peer Object	Function Code (Binary)	Resulting COB-ID	Communication Parameters at Index
Emergency	0001	129 (81h) – 255 (FFh)	1014h, 1015h
PDO1 (Tx)	0011	385 (181h) – 511 (1FFh)	1800h
PDO1 (Rx)	0100	513 (201h) – 639 (27Fh)	1400h
PDO2 (Tx)	0101	641 (281h) – 767 (2FFh)	1801h
PDO2 (Rx)	0110	769 (301h) – 895 (37Fh)	1401h
PDO3 (Tx)	0111	897 (381h) – 1023 (3FFh)	1802h
PDO3 (Rx)	1000	1025 (401h) – 1151 (47Fh)	1402h
PDO4 (Tx)	1001	1153 (481h) – 1279 (4FFh)	1803h
PDO4 (Rx)	1010	1281 (501h) – 1407 (57Fh)	1403h
SDO (Tx)	1011	1409 (581h) – 1535 (5FFh)	1200h
SDO (Rx)	1100	1537 (601h) – 1663 (67Fh)	1200h
NMT Error Control	1110	1793 (701h) – 1919 (77Fh)	1016h, 1017h

The CANopen NIM's Object Dictionary

About the Object Dictionary

The object dictionary is the most important part of the CANopen device model *(see page 65)* because it is a map to the internal structure of CANopen devices (according to CANopen profile DS-401). A given device's object dictionary is a lookup table that describes the data types, COBs, and application objects the device uses.

By accessing a particular device's object dictionary structure through the CANopen fieldbus, you can predict its network behavior and, therefore, build a distributed application that implements it.

Index Ranges

CANopen addresses the contents of the object dictionary using a16-bit index with an 8-bit subindex. There are three object dictionary regions:

Index (hex)	Object	Function
1000-1FFF	communication profile area	communication capabilities
2000-5FFF	manufacturer-specific area	diagnostic information, some I/O data
6000-9FFF	device-specific profile area	I/O data

Manufacturer-specific objects and device-specific objects are mappable to PDOs, which are then sent along the CANopen fieldbus.

Standard Device Profiles

Profiles for the standard devices that the CANopen NIM supports are described in the following tables.

Digital Inputs

When an 8-bit digital input for a digital I/O module is changed, a default TxPDO is transmitted.

Index	Subindex	Name	Type	Attr.	Default	Description
6000h	0	8-bit digital input	unsigned8	ro	none	number of digital input blocks
	1	input block	unsigned8	ro	none	1. digital input block (8 digital input channels from left to right, starting at the NIM)
	2	input block	unsigned8	ro	none	2. digital input block (next 8 digital input channels from left to right)
	. . .	. . .	. . .	. . .	. . .	. . .
	0x20	input block	unsigned8	ro	none	32. digital input block

Digital Outputs

The 8-bit digital output of a digital I/O module is asynchronously received.

Index	Subindex	Name	Type	Attr.	Default	Description
6200h	0	8-bit digital output	unsigned8	ro	none	number of digital output blocks
	1	output block	unsigned8	rw	none	1. digital input block (8 digital output channels from left to right, starting at the NIM)
	2	output block	unsigned8	rw	none	2. digital input block (next 8 digital output channels from left to right)
	. . .	. . .	. . .	. . .	. . .	. . .
	0x20	output block	unsigned8	rw	none	32. digital output block

Analog Inputs

The 16-bit analog input default value is 0 (no channels selected).

Index	Subindex	Name	Type	Attr.	Default	Description
6401h	0	16-bit analog input	unsigned8	ro	none	number of analog input channels
	1	channel	unsigned16	ro	none	1. analog 16-bit input (input channels from left to right, starting at the NIM)
	. . .	. . .	. . .	. . .	. . .	. . .
	0x20	channel	unsigned16	ro	none	32. analog 16-bit input

Analog Outputs

The 16-bit analog output default value is 0 (no channels selected).

Index	Subindex	Name	Type	Attr.	Default	Description
6411h	0	16-bit analog output	unsigned8	ro	none	number of analog output channels
	1	1. channel	unsigned16	rw	none	1. analog 16-bit output (output channels from left to right, starting at the NIM)
	. . .	. . .	. . .	. . .	. . .	. . .
	0x20	channel	unsigned16	rw	none	32. analog 16-bit output

Manufacturer-Specific Objects

Profiles for the manufacturer-specific devices that the CANopen NIM supports are described in the following tables.

Analog Global Interrupt Enable

Analog TxPDO transmissions need to be enabled by object 6423, the object that determines the transmission of analog input values. Since the default value is *false*, no analog input objects are transmitted. To enable transmission, set this object to *true* by writing *1* to index 6423.

Index	Subindex	Name	Data Type	Attr.	Default	Description
6423h	0	analog global interrupt enable	boolean	rw	FALSE	determines the transmission of analog input values

NOTE: According to CANopen specification DS-401, the STB NCO 2212 CANopen NIM will not be able to transmit an analog TxPDO unless the transmission is enabled by writing *1* to index 6423.

Mandatory CANopen Entries

All nodes in a CANopen-compliant network must support the mandatory entries in the following table.

Index	Subindex	Name	Data Type	Attr.	Default	Description
1000h	0	device type information	unsigned32	ro	none	device type
1001h	0	error register	unsigned32	rw	0	error register
1018h		identity object				identity object
	0	= 4 (number of subindex entries)	unsigned8	ro	none	number of subindex entries (4)
	1	vendor ID	unsigned32	ro	none	vendor ID
	2	product code	unsigned32	ro	none	product code
	3	revision number	unsigned32	ro	none	revision number
	4	serial number	unsigned32	ro	none	serial number

Remote Virtual Placeholder Objects

When you enable the remote virtual placeholder configuration option *(see page 173)*, 4 additional objects appear in the object dictionary. If this option is not enabled, these objects are not present. None of these 4 objects is PDO mappable.

Index	Subindex	Name	Description	Data Type	Attr.	Default
4200h	0	IOC	Island operation control	unsigned16	rw	0
4201h	0	IOS	Island operation status	unsigned16	ro	
4202h		VPCW	Virtual placeholder configuration write			
	0		Largest subindex	unsigned8	ro	2
	1		Desired virtual placeholder configuration for island addresses 32 ... 1	unsigned32	wo	0
	2		Desired virtual placeholder configuration for island addresses 64 ... 33	unsigned32	wo	0 (always 0 for the standard NIM)
4203h		VPCR	Virtual placeholder configuration read			
	0		Largest subindex	unsigned8	ro	2
	1		Actual virtual placeholder configuration for island addresses 32 ... 1	unsigned32	ro	
	2		Actual virtual placeholder configuration for island addresses 64 ... 33	unsigned32	ro	

These four object are described in more detail in *Special Objects for the Remote Virtual Placeholder Option, page 177*.

Object Descriptions and Index Addresses

Introduction

A COB is a unit of transportation, or *message*, in a CAN network. Data on a CAN network must be sent in COBs. A single COB can contain at most 8 bytes of data. There are 2048 different COB-IDs in a CAN network.

Descriptions and index addresses (in the NIM's object dictionary) of the most commonly used Advantys STB COB-IDs follow.

- communications objects
- manufacturer-specific objects
- device-specific objects

Communication Objects

There are various types of communications objects within the CANopen network protocol.

CANopen specifies two mechanisms for data exchange:

- *process data objects*—PDOs are transmitted as unconfirmed broadcast messages or sent from a *producer* device to a *consumer* device. The TxPDO from the producer device has a specific identifier that corresponds to the RxPDO of the consumer devices.
  These messages have a maximum of 8 bytes per PDO. They are used for real-time data exchange. Data contained in synchronous PDOs may either be predefined by the device manufacturer or configured with the application.
- *service data objects*—SDOs are used by the CANopen master to access (read/write) the object dictionaries of network nodes. In some networks, asynchronous SDOs can be used to alter the identifier allocation with configuration software.

CANopen specifies two services for network management:

- *special function objects*—These protocols provide application-specific network synchronization and emergency message transmission.
- *network management*—NMT protocols provide services for network initialization, error control, and device status control.

Supported Communication Objects

The following table lists those objects that the Advantys STB CANopen NIM supports:

Index	Object	Name	Type	Acct.	M/O*
1000	variable	device type	unsigned32	ro	M
1001	variable	error register	unsigned8	ro	M
1003	array	predefined error field	unsigned32	ro	O

Index	Object	Name	Type	Acct.	M/O*
1005	variable	COB-ID SYNC message	unsigned32	rw	O
1008	variable	manufacturer device name	vis. string	c	O
100C	variable	guard time	unsigned32	rw	O
100D	variable	life time factor	unsigned32	rw	O
1010	variable	store parameters	unsigned32	rw	O
1011	variable	restore default parameters	unsigned32	rw	O
1014	variable	COB-ID emergency	unsigned32	rw	O
1016	array	consumer heartbeat time	unsigned32	rw	O
1017	variable	producer heartbeat time	unsigned16	rw	O
1018	record	identity object	identity	ro	M
. . .	. . .	. . .	. . .	. . .	. . .
11FF	reserved				
*M = *mandatory*, O = *optional*					

Detailed descriptions of the individual COBs in the above table follow.

Device Type

The device type COB describes the type of device and its functionality. It is composed of a 16-bit field that describes the employed device profile:

Index	Subindex	Name/Purpose	Data Type	Attr.
1000h	0	device type	unsigned32	ro

A second 16-bit field gives additional information about the device's optional functionality:

Additional Information (MSB)	Device Profile (DS-401) (LSB)
0000 0000 0000 wxyz	0401
Note: z = 1 (digital input), y = 1 (digital output), x = 1 (analog input), w = 1 (analog output)	

For multiple-device modules, the index of the *additional information* parameter is FFFFh. The device profile number referenced by object 1000 is that of the first device in the object dictionary. All other devices of a multiple-device module identify their profiles as objects 67FFh + x * 800h (x = internal number of the device, 0 to 7).

This object will be dynamically generated at startup, since the device type depends on the actual island configuration.

Error Register

Devices map any internal errors to the error register byte:

Index	Subindex	Name/Purpose	Data Type	Attr.
1001h	0	error register	unsigned8	ro

This *error register* entry is mandatory for all devices. It is part of the emergency object.

Predefined Error Field

The predefined error field COB holds errors that occur on the device that have been signaled via the emergency object, providing an error history:

Index	Subindex	Name/Purpose	Data Type	Attr.
1003h	-	predefined error field (error history)		
	0	number of errors	unsigned8	rw
	1	actual error	unsigned32	rw
	2 . . . 10	error field	unsigned32	rw

The entry at subindex 0 contains the number of actual errors that are recorded in the array starting at subindex 1. Every new error is stored at subindex 1, pushing older errors down the list. Writing *0* to subindex 0 will empty the array, deleting the entire error history. Errors numbers (of type unsigned32) are composed of 16-bit error codes and an additional, manufacturer-specific, 16-bit error information field.

The error code is contained in the lower 2 bytes (LSB) and the additional information is included in the upper 2 bytes (MSB):

Additional Information (MSB)	Error Code (LSB)

COB-ID SYNC Message

The COB-ID SYNC message COB at index 1005h defines the COB-ID of the synchronization object (SYNC). (It does not generate SYNC messages.) It also defines whether the device generates the SYNC.

Index	Subindex	Name/Purpose	Data Type	Attr.
1005h	0	COB-ID SYNC message	unsigned32	rw

The default value is 0x0000 0080.

Manufacturer Device Name

The manufacturer device name COB represents the strings for the CANopen NIM:

Index	Subindex	Name/Purpose	Data Type	Attr.
1008h	0	manufacturer device name	ASCII string	c

Guard Time

The user can adjust the guard time with the COB at index 100Ch:

Index	Subindex	Name/Purpose	Data Type	Attr.
100Ch	0	guard time (default = 0; unused)	unsigned16	rw

Life Time Factor

The user can adjust the life time with the COB at index 100Dh:

Index	Subindex	Name/Purpose	Data Type	Attr.
100Dh	0	life time factor (default = 0; unused)	unsigned8	rw

Store Parameters

By writing the ASCII string *save* (hex code 0x65766173) to the store parameters COB, all NIM parameters are stored in Flash memory:

Index	Subindex	Name/Purpose	Data Type	Attr.
1010h	-	store parameters	-	-
	0	largest subindex: 2	unsigned8	ro
	1	store all parameters	unsigned32	rw

Subindex 1 refers to index 1000h through 1FFFh and 6423h. This is allowed only in the pre-operational state. Otherwise, SDO access is aborted. As a consequence, the micro controller is busy for a few seconds with Flash programming (an exclusive action). During this time there is no communication on either the fieldbus or island bus.

Restore Default Parameters

By writing the ASCII string *load* (hex code 0x64616F6C) to the restore default parameters COB, the NIM's default parameters are restored:

Index	Subindex	Name/Purpose	Data Type	Attr.
1011h	-	restore default parameters	-	-
	0	largest subindex: 1	unsigned8	ro
	1	store all parameters	unsigned32	rw

Subindex 1 refers to index 1000h through 1FFFh and 6423h. This is allowed only in the pre-operational state. Otherwise, SDO access is aborted. As a consequence, the micro controller is busy for a few seconds with Flash programming (an exclusive action). During this time there is no communication on either the fieldbus or island bus.

COB-ID Emergency Message

The COB-ID emergency message COB uses CANopen's default:

Index	Subindex	Name/Purpose	Data Type	Attr.
1014h	0	COB-ID emergency message (default = 0x0000 0080 + node ID)	unsigned32	rw

Consumer Heartbeat Time

The consumer heartbeat time COB defines the expected heartbeat cycle time and, therefore, has to be longer than the corresponding time configured for the heartbeat of the producing device:

Index	Subindex	Name/Purpose	Data Type	Attr.
1016h	-	consumer heartbeat time		
	0	number of entries: 1	unsigned8	ro
	1	see below (default = 0; not used)	unsigned32	rw

Monitoring starts after the reception of the first heartbeat. The heartbeat time has to be a multiple of 1 ms:

Reserved (MSB)	Node ID	Heartbeat Time (LSB)
—	unsigned8	unsigned16

Producer Heartbeat Time

The producer heartbeat time COB defines the cycle time of the heartbeat. If it is not used, the producer heartbeat time is 0. The time has to be a multiple of 1 ms.

Index	Subindex	Name/Purpose	Data Type	Attr.
1017h	0	producer heartbeat time (default = 0; unused)	unsigned16	rw

Identity Object

The identity object (index 1018h) COB contains general information about the NIM:

Index	Subindex	Name/Purpose	Data Type	Attr.
1018h	-	identity object (contains general device (NIM) information)	-	-
	0	number of entries: 3	unsigned8	ro
	1	vendor ID code	unsigned32	ro
	2	product code: 33001546 (Standard)	unsigned32	ro
	3	major and minor product revision number	unsigned32	ro

The vendor ID code (subindex 1) contains the unique value allocated to Schneider Electric. The product code (subindex 2) is a unique number that determines the product within Schneider. The revision number (subindex 3) consists of a major revision number and a minor revision number. The major revision number identifies a specific CANopen behavior. When the CANopen functionality is expanded, the major revision has to be incremented. The minor revision number identifies different versions with the same CANopen behavior.

Mandatory CANopen Objects

There are objects that every CANopen node is required to support. Mandatory COBs are specified in CiA DS-301. The following tables present detailed descriptions and index addresses of those objects.

Server SDO Parameters

The server SDO parameters COB uses CANopen's default:

Index	Subindex	Name/Purpose	Data Type	Attr.
1200h	-	server SDO parameters	unsigned8	-
	0	number of entries: 2	unsigned32	ro
	1	COB-ID client . . . server (Rx) default = 0x0000 0600 + node ID	unsigned32	ro
	2	COB-ID server . . . client (Tx) default = 0x0000 0580 + node ID	unsigned32	ro

RxPDO Communication Parameters

The RxPDO communication parameters COB contains the communication parameters for those PDOs that the device is able to receive:

Index	Subindex	Name/Purpose	Data Type	Attr.
1400h . . . 141Fh	-	RxPDO communication parameter (PDO1) . . . RxPDO communication parameter (PDO32)	-	-
	0	number of entries: 2	unsigned8	ro
	1	COB-ID of the RxPDO1 . . . RxPDO32 default = 0x0000 0200 + node ID for 1400 default = 0x0000 0300 + node ID for 1401 default = 0x0000 0400 + node ID for 1402 default = 0x0000 0500 + node ID for 1403 default = 0x8000 0000 (not used) for 1404...141F	unsigned32	rw
	2	transmission type of RxPDO1; default = 255	unsigned8	rw

RxPDO Mapping Parameters

The RxPDO mapping parameters (for PDO1 to PDO32) COBs can be found in 1600h through 161Fh. This object contains the mappings for those PDOs that the device is able to receive. Subindex 0 contains the number of valid entries within the mapping record.

Index	Subindex	Name/Purpose	Data Type	Attr.
1600h	-	RxPDO mapping parameter for PDO1	-	-
	0	number of entries: 0 . . . 8	unsigned8	rw
	1	mapped object, index, subindex, bit length (default = 0x6200 0108)	unsigned32	rw
	2	mapped object, index, subindex, bit length (default = 0x6200 0208)	unsigned32	rw
	. . .	. . .	. . .	. . .
	8	mapped object, index, subindex, bit length (default = 0x6200 0808)	unsigned32	rw

NOTE: The NIM provides the default PDO mapping (according to CANopen specification DS-401) for PDO1 through PDO4. Default entries depend on the island configuration and are dynamically entered to subindexes 1 through 8. When the appropriate objects are present in the object dictionary, the default values are set accordingly. Otherwise the default entries are 0000.

TxPDO Communication Parameters

The TxPDO communication parameters COB contains the communication parameters for those PDOs that the device is able to transmit:

Index	Subindex	Name/Purpose	Data Type	Attr.
1800h . . . 181Fh		TxPDO comm. parameter (PDO1) . . . TxPDO comm. parameter (PDO32)	-	-
	0	number of entries: 3	unsigned8	ro
	1	COB-ID of the TxPDO1 . . . TxPDO32 default = node 0x0000 0180 + node ID for 1800 default = node 0x0000 0280 + node ID for 1801 default = node 0x0000 0380 + node ID for 1802 default = node 0x0000 0480 + node ID for 1803 default = node 0x8000 0000 (not used) for 1804 through 181F	unsigned32	rw
	2	transmission type of TxPDO1 (default = 255)	unsigned8	rw
	3	inhibit time (default = 0)	unsigned16	rw

TxPDO Mapping Parameter for PDO1

The TxPDO mapping parameter for PDO1 COB contains mappings for those PDOs that the device is able to transmit. The subindex 0 contains the number of valid entries within the mapping record. Default PDO mapping (according to CANopen specification DS-401) is provided by the NIM for PDO1 through PDO4. The default entries depend on the island configuration and are dynamically entered into subindexes 1 through 8. When the appropriate objects are present in the object dictionary, the default values are set accordingly. Otherwise the default entries are 0000:

Index	Subindex	Name/Purpose	Data Type	Attr.
1A00h	-	TxPDO mapping parameter for PDO1	-	-
	0	number of entries: 0 . . . 8	unsigned8	rw
	1	mapped object, index, subindex, bit length (default - 0x6000 0108)	unsigned32	rw
	2	mapped object, index, subindex, bit length (default - 0x6000 0208)	unsigned32	rw
	. . .	. . .	. . .	. . .
	8	mapped object, index, subindex, bit length (default - 0x6000 0808)	unsigned32	rw

Manufacturer-Specific Objects

Objects in the following tables fall in the index range that CANopen reserves for manufacturer-specific objects (DS-301). These objects contain special modules and some manufacturer-specific items, including some diagnostic information.

Manufacturer-specific objects are in the index range 2000h to 5FFFh. The CANopen NIM supports the following objects:

Index	Subindex
2000h . . . 2xxxh	a list of special input objects that can not be identified by the NIM because they are not in DS-401 or DSP-402 supported object lists
3000h . . . 3xxxh	a list of special output objects that can not be identified by the NIM because they are not in DS-401 or DSP-402 supported object lists
4000h . . . 4xxxh	communication diagnostics support objects

Those objects that can not be identified because they are not in DS-401 or DS-402 object lists are sorted according to object type and length, according to the following algorithm:

Type	Length	Index Lists	Data Type	Attr.
input	1 byte	2000h . . .	unsigned8	ro
input	2 byte	2200h . . .	unsigned16	ro
input	3 byte	2400h . . .	unsigned24	ro
input	4 byte	2600h . . .	unsigned32	ro
input	5 byte	2800h . . .	unsigned40	ro
input	6 byte	2A00h . . .	unsigned48	ro
input	7 byte	2C00h . . .	unsigned56	ro
input	8 byte	2E00h . . .	unsigned64	ro
output	1 byte	3000h . . .	unsigned8	rw
output	2 byte	3200h . . .	unsigned16	rw
output	3 byte	3400h . . .	unsigned24	rw
output	4 byte	3600h . . .	unsigned32	rw
output	5 byte	3800h . . .	unsigned40	rw
output	6 byte	3A00h . . .	unsigned48	rw
output	7 byte	3C00h . . .	unsigned56	rw
output	8 byte	3E00h . . .	unsigned64	rw

These lists are set up dynamically at startup, depending on the availability of special objects. Objects of the same type are listed at subindex 0 of a subsequent index.

Two-byte data sent from the HMI to the PLC will be put in the 2200 object list. Two-byte data sent from the PLC to the HMI will be put in the 3200 object list.

Global Bits

Each of the 16 bits in the global bits manufacturer-specific object indicates a specific error on the island bus:

Index	Subindex	Name/Purpose	Data Type	Attr.
4000h	0	global bits	unsigned16	r0

Errors marked with an asterisk (*) in the global bits table are fatal NIM errors. They are caused by internal errors related to either the NIM or a failure in the island configuration software or hardware:

Bit	Meaning
D0*	fatal error—Because of the severity, no further communications are possible on the island bus.
D1*	module ID error—A standard CANopen device is using a module ID reserved for the Advantys STB modules.
D2*	Auto-addressing has failed.
D3*	Mandatory module configuration error.
D4*	process image error—Either the process image configuration is inconsistent or it could not be set during auto-configuration.
D5*	auto-configuration error—A module has been detected out of order and the NIM can not complete auto-configuration.
D6	Island bus management error detected by the NIM.
D7*	assignment error—The initialization process in the NIM has detected a module assignment error.
D8*	internal triggering protocol error
D9*	module data length error
D10*	module configuration error
D11 ... D15	reserved
*fatal NIM errors The detection of these errors will result in the stopping of the island bus. The only ways to get out of this error state are to cycle the power or reset the island.	

Communication Diagnostics

The communication diagnostic object represents the main states of the island bus scanner, which is the firmware that drives the island bus. This word is divided into a low byte (D0–D7), representing the island bus state, and a high byte (D8 through D15) that contains the communication diagnostics:

Index	Subindex	Name/Purpose	Data Type	Attr.
4001h	0	island bus state/communication diagnostics	unsigned16	r0

The following low-byte values are possible for the communication diagnostic manufacturer-specific object:

Byte Value	Meaning
00h	The island is initializing
40h	The island bus has been set to pre-operational mode, for example, by the reset function in the Advantys STB configuration software.
60h	NIM is configuring or auto-configuring—Communication to all modules is reset.
61h	NIM is configuring or auto-configuring—Checking the module ID.
62h	The NIM is auto-addressing the island.
63h	NIM is configuring or auto-configuring—Bootup is in progress.
64h	The process image is being set up.
80h	Initialization is complete, the island bus is configured, the configuration matches, and the island bus is not started.
81h	configuration mismatch—Non-mandatory or unexpected modules in the configuration do not match, and the island bus is not started.
82h	configuration mismatch—At least one mandatory module does not match, and the island bus is not started.
83h	serious configuration mismatch—The island bus is set to pre-operational mode and initialization is aborted.
A0h	The configuration matches, and the island bus is operating.
A1h	Island is operational with a configuration mismatch. At least one standard module does not match, but all the mandatory modules are present and operating.
A2h	serious configuration mismatch—The island bus was started but is now in pre-operational mode because of one or more mismatched mandatory module(s).
C0h	Island has been set to pre-operational mode.

The following high-byte values are possible for the communication diagnostic manufacturer-specific object. Errors marked with an asterisk (*) in the communication diagnostic tables are fatal NIM errors. They are caused by internal errors related to either the CANopen controller or a failure in the island configuration software or hardware:

Communication Diagnostic	Meaning of Value
D8*	1 = low-priority receive queue software overrun error.
D9*	1 = NIM overrun error.
D10*	1 = island bus-off error.
D11	1 = error counter in NIM has reached the warning level and the error status bit has been set.

Communication Diagnostic	Meaning of Value
D12	1 = NIM error status bit has been reset.
D13*	1 = low-priority transfer queue software overrun error.
D14*	1 = high-priority receive queue software overrun error.
D15*	1 = high-priority transfer queue software overrun error.
*fatal NIM errors	

The detection of these errors will result in the stopping of the island bus. After a 5-second pause, the NIM will initiate a restart.

Node Configured

The node configured manufacturer-specific object is a bit field:

Index	Subindex	Name/Purpose	Data Type	Attr.
4002h	-	node configured	-	-
	0	number of entries = 8	unsigned8	ro
	1	module 16 . . . 1	unsigned16	ro
	2	module 32 . . . 17	unsigned16	ro
	3	module 48 . . . 33	unsigned16	ro
	4	module 64 . . . 49	unsigned16	ro
	5	module 80 . . . 65	unsigned16	ro
	6	module 96 . . . 81	unsigned16	ro
	7	module 112 . . . 97	unsigned16	ro
	8	module 127 . . . 113	unsigned16	ro

Each bit represents one specific module (node) on the island bus. When a module is configured, the corresponding bit is set.

Node Operational

The node operational manufacturer-specific object is a bit field:

Index	Subindex	Name/Purpose	Data Type	Attr.
4003h	-	node operational	-	-
	0	number of entries = 8	unsigned8	ro
	1	module 16 . . . 1	unsigned16	ro
	2	module 32 . . . 17	unsigned16	ro
	3	module 48 . . . 33	unsigned16	ro
	4	module 64 . . . 49	unsigned16	ro
	5	module 80 . . . 65	unsigned16	ro
	6	module 96 . . . 81	unsigned16	ro
	7	module 112 . . . 97	unsigned16	ro
	8	module 127 . . . 113	unsigned16	ro

Each bit represents one specific module (node) on the island bus. If a module is set to operational, the corresponding bit is set.

Node Error

The node error manufacturer-specific object is a bit field:

Index	Subindex	Name/Purpose	Data Type	Attr.
4004h	-	node error	-	-
	0	number of entries = 8	unsigned8	ro
	1	module 16 . . . 1	unsigned16	ro
	2	module 32 . . . 17	unsigned16	ro
	3	module 48 . . . 33	unsigned16	ro
	4	module 64 . . . 49	unsigned16	ro
	5	module 80 . . . 65	unsigned16	ro
	6	module 96 . . . 81	unsigned16	ro
	7	module 112 . . . 97	unsigned16	ro
	8	module 127 . . . 113	unsigned16	ro

Each bit represents one specific module (node) on the island bus. After the master receives an emergency message (not error-free) from a module, the corresponding bit is set.

Node Assembly Fault

The node assembly fault manufacturer-specific object is a bit field:

Index	Subindex	Name/Purpose	Data Type	Attr.
4005h	-	node assembly fault	-	-
	0	number of entries = 8	unsigned8	ro
	1	module 16 . . . 1	unsigned16	ro
	2	module 32 . . . 17	unsigned16	ro
	3	module 48 . . . 33	unsigned16	ro
	4	module 64 . . . 49	unsigned16	ro
	5	module 80 . . . 65	unsigned16	ro
	6	module 96 . . . 81	unsigned16	ro
	7	module 112 . . . 97	unsigned16	ro
	8	module 127 . . . 113	unsigned16	ro

Each bit represents one specific module (node) on the island bus. If the configuration of a module mismatches, the corresponding bit is set.

NIM Status

The NIM status COB describes the status of the CANopen NIM:

Index	Subindex	Name/Purpose	Data Type	Attr.
4006h	0	NIM status	unsigned16	ro

Errors marked with an asterisk (*) in the NIM status table are fatal NIM errors. They are caused by internal errors related to either the NIM or a failure in the island configuration software or hardware:

Bit	Meaning of Value
D0 ... D7	reserved
D8	module failure—Bit 0 is set to 1 if any module on the island bus fails.
D9	internal failure (value of 1)—At least one *global bit* was set (except *RESET*). When one of these bits is set, bit D4 of object 1003h is also set.
D10	external failure (value of 1)—The problem is on the fieldbus.
D11, D12	reserved
D13	reflex action (value of 1)—reflex action functionality has been configured. (For NIMs with firmware version 2.0 or greater.)
D14	hot-swapped maodules (value of 1)— One or more island modules have been hot-swapped. (For NIMs with firmware version 2.0 or greater.)
D15	reserved

The detection of these errors will result in the stopping of the island bus. After a 5 s pause, the NIM will initiate a restart.

Device-Specific Objects

Device-specific objects contain the bulk of the process I/O data. These objects are in the index range 6000h to 9FFFh:

Index	Subindex	Name/Purpose	Data Type	Attr.
6000h	0	number of input 8 bit	unsigned8	ro
	1	first island object 6000	unsigned8	ro
	. . .	. . .		
6200h	0	number of output 8 bit	unsigned8	rw
	1	first island object 6200	unsigned8	rw
	. . .	. . .		

NOTE: Mapped objects of island modules should be according to CANopen device profiles DS-401 (I/O modules) and DSP-402 (drives).

The following objects are supported in device profile DS-401 (I/O modules):

Index/Subindex	Input	Index/Subindex	Output
6000/1 . . . 254	digital in (8-bit)	6200/1 . . . 254	digital out (8-bit)
6100/1 . . . 254	digital in (16-bit)	6300/1 . . . 254	digital out (16-bit)
6120/1 . . . 254	digital in (32-bit)	6320/1 . . . 254	digital out (32-bit)
6400/1 . . . 254	analog in (8-bit)	6400/1 . . . 254	analog out (8-bit)
6401/1 . . . 254	analog in (16-bit)	6411/1 . . . 254	analog out (16-bit)
6402/1 . . . 254	analog in (32-bit)	6412/1 . . . 254	analog out (32-bit)
6403/1 . . . 254	analog in (float)	6413/1 . . . 254	analog out (float)

These objects are the true input and output data. Besides these, there are several objects defined in DS-401 that are treated as parameters and assumed to be unmapped.

These objects are listed in the object dictionary with the same indexes (under subsequent subindexes). Subindex 0 has been added to display the number of entries.

SDO Services

SDOs are the mechanisms for establishing a client/server relationship between CANopen devices. They are used by the CANopen master to access object dictionaries of network nodes. There are two types of SDOs implemented in CANopen devices:

- *server SDO*—Each CANopen device is required to allow access to its own object dictionary through at least one server SDO.
- *client SDO*—A client SDO can read from and write to the object dictionary of a server device.

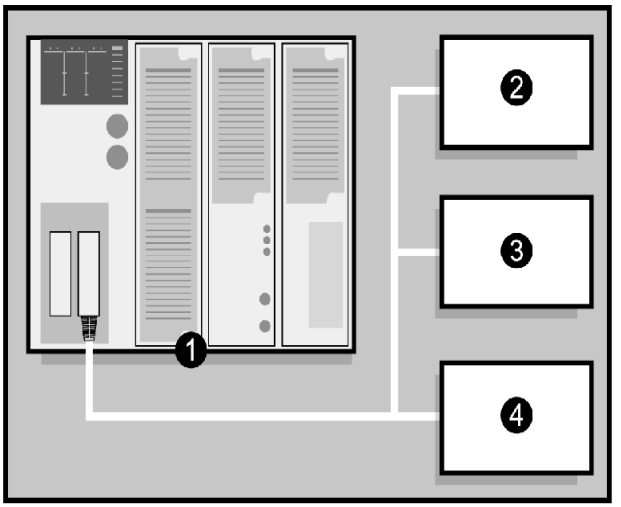
Each SDO has two message identifiers that indicate the direction of travel (upload/download) in SDO transfers:

- *SDO upload*—Messages transmitted from the client to the server are SDO upload messages.
- *SDO download*—Messages transmitted from the server to the client are SDO download messages.

The SDO transfer procedure employs one of three domain protocols, depending on the particular nature and size of the data transfer:

- The *expedited download/upload* domain protocol is implemented for devices that support objects that are not larger than 4 bytes.
- The *segmented download/upload* domain protocol is implemented for devices that support objects larger than 4 bytes. The complete data is transferred in a series of confirmed 4-byte segments.

The implementation of SDO transmission and reception types on a CANopen network are shown in the following figure:



1 CANopen master—The master sequentially transmits SDO requests to nodes using CAN ID 600h + node ID. Expected replies use CAN ID 580h + node ID.
2 Node 1—Node 1 receives SDO 601h (600h + node ID) and replies with SDO 581h (580 + node ID).
3 Node 2—Node 2 receives SDO 602h and replies with SDO 582h.
4 Node 3—Node 3 receives SDO 603h and replies with SDO 581h.

PDO Mapping

CANopen and PDOs

Transmitted as broadcast messages, process data objects (PDOs) are unconfirmed messages used for real-time data exchange of short blocks of high-priority data. A special feature of CANopen is that data contained in PDOs may be either predefined by the device manufacturer or configured by the application.

Each of the 8 bytes (or fewer) in a PDO is defined through mapping information stored in the object dictionary of its producer and consumer devices.

PDO types

PDO usage is based on CANopen's producer/consumer model. A PDO's designation as either *transmit* or *receive* is relative to the nature of each device, depending on how the same identifier (signal value) has been mapped by those devices. If a device produces a PDO, the PDO is a *transmit* PDO (TxPDO) of that device. If a device consumes a PDO, it is a *receive* PDO (RxPDO) of that device.

Predefined Connection Set

CANopen's predefined connection set allows for peer-to-peer communications between a master device and its nodes without requiring an identifier distribution process:

Object	Function Code (Binary)	COB-ID	Comm. Parameters at Index
emergency	0001	129 (81h)–255 (2FFh)	1014h. 1015h
PDO1 (Tx)	0011	385 (181h)–511 (1FFh)	1800h
PDO1 (Rx)	0100	513 (201h)–639 (639h)	1400h
PDO2 (Tx)	0101	641 (281h)–767 (2FFh)	1801h
PDO2 (Rx)	0110	769 (301h)–895 (37Fh)	1401h
PDO3 (Tx)	0111	897 (381h)–1023 (3FFh)	1802h
PDO3 (Rx)	1000	1025 (401h)–1151 (47Fh)	1402h
PDO4 (Tx)	1001	1153 (481h)–1279 (4FFh)	1803h
PDO4 (Rx)	1010	1281 (501h)–1407 (57Fh)	1403h
SDO (Tx)	1011	1409 (581h)–1535 (5FFh)	1200h
SDO (Rx)	1100	1537 (601h)–1663 (67Fh)	1200h
NMT error control	1110	1793 (701h)–1919 (77Fh)	1015h, 1017h

The predefined connection set also reserves message identifiers for broadcast messages:

Object	Function Code (Binary)	COB-ID	Comm. Parameters at Index
NMT	0000	0	
SYNC	0001	128 (80h)	1005h, 1006h, 1007h

PDO Mapping Table

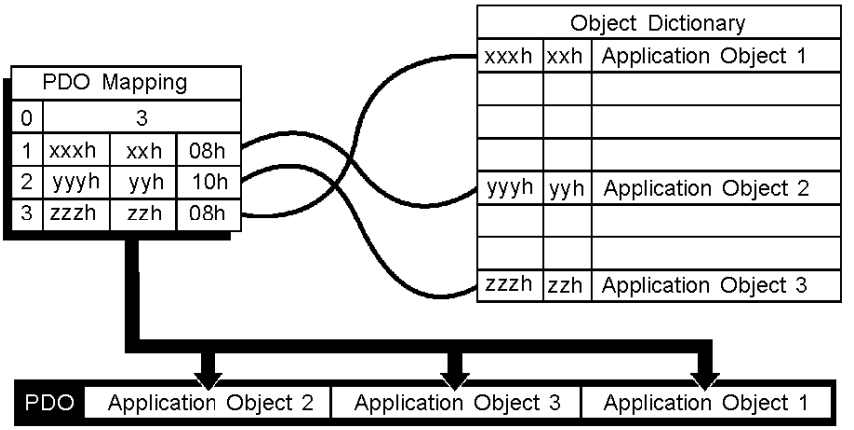
Predefined PDO mappings for different Advantys STB nodes are outlined in the following table.

PDO	Description
RxPDO1	asynchronously receives 8 bytes, object index 6200, subindex 1 . . . 8 (digital output data)
RxPDO2	asynchronously receives four 16-bit values, object index 6411, subindex 1 . . . 4 (analog output data)
RxPDO3	asynchronously receives four 16-bit values, object index 6411, subindex 5 . . . 8 (analog output data)
RxPDO4	asynchronously receives four 16-bit values, object index 6411, subindex 9 . . . 12 (analog output data)
TxPDO1	event-driven transmission of 8 bytes, object index 6000, subindex 1 . . . 8 (digital input data)
TxPDO2	event-driven transmission of four 16-bit values, object index 6401, subindex 1 . . . 4 (analog input data)
TxPDO3	event-driven transmission of four 16-bit values, object index 6401, subindex 5 . . . 8 (analog input data)
TxPDO4	event-driven transmission of four 16-bit values, object index 6401, subindex 9 . . . 12 (analog input data)

Mapping to Application Objects

PDO mapping information (part of the object dictionary) describes the arrangement of the application objects to a PDO.

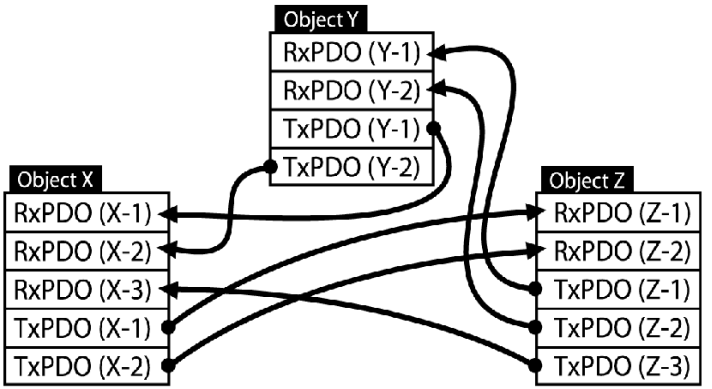
PDO mapping information describes the arrangement of application objects to a PDO. The NIM starts with default mapping corresponding to DS-401:



NOTE: Sub-index 0 indicates the number of mapped objects that follow in the object list.

The STB NCO 2212 CANopen NIM also supports variable (dynamic) mapping. With variable mapping, users can instruct the master to reassign RxPDOs and TxPDOs implemented with the node's object dictionary entries. In this way, nodes can be configured to use specific CAN identifiers for TxPDOs while listening for specific CAN identifiers with RxPDOs. (You will have to configure the corresponding TxPDOs and RxPDOs for the intended objects in the object dictionary mapping table.)

Variable PDO mapping among theoretical objects X, Y, and Z is shown in the following figure:



Network Management

Summary

CANopen uses a node-oriented NMT structure that follows a master/slave model. This structure requires one device on the network to function as the NMT master, with other nodes acting as its slaves.

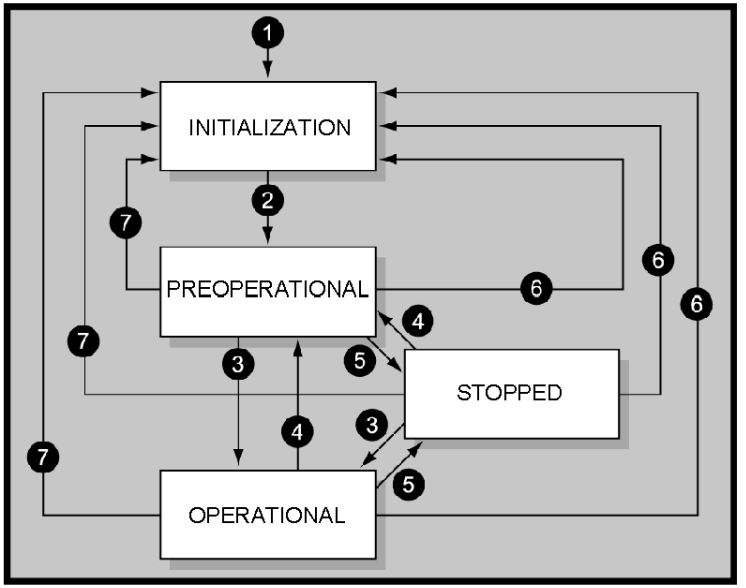
CANopen NMT provides these functionality groups:

- *module control services*—initialization of those NMT slaves that will be implemented in the distributed application
- *error control services*—supervision of nodes and the network's communication status
- *configuration control services*—uploading/downloading configuration data to or from a module on the network

An NMT slave represents that part of a node that is responsible for its NMT functionality. The NMT slave is identified by its unique module ID.

State Machine

CANopen NMT slave devices use the commissioning state machine to describe the sequence for powering up and initializing devices to their preoperational, operational, or support states:



1 At power up, the node's initialization state is entered autonomously.
2 After initialization, the preoperational state is entered automatically.
3 START_REMOTE_NODE indication
4 Enter_PRE-OPERATIONAL_State indication
5 STOP_REMOTE_NODE indication
6 RESET_NODE indication
7 RESET_COMMUNICATION indication

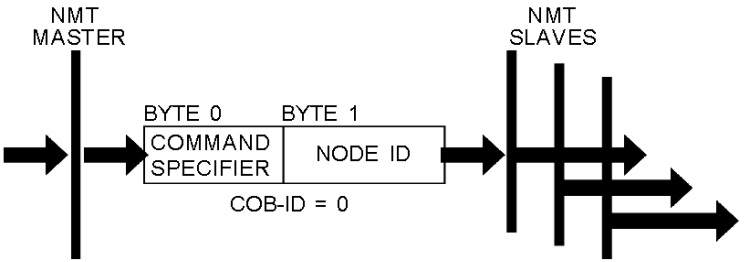
After initialization, the device can be in one of three states:

- *preoperational state*—In this state, you can configure the node with an SDO, although PDO communication is not allowed.
- *operational state*—In this state, all COBs are active. SDO access to the object dictionary is possible.
- *stopped state*—When the device is switched to this state, SDO and PDO communications cease.

Each state indicates those commands the node will accept from the NMT master.

State Switching

The figure below shows the structure of a state transition message sent from the NMT master to all nodes (COB-ID = 0):



SYNC Messages

Introduction

SYNC messages are broadcast periodically on the network by a synchronization device. Using the SYNC message, devices on the CANopen network can be synchronized to implement coordinated data acquisition mechanisms. Whether an object uses the SYNC event dictates its transmission mode.

Transmission Modes

A PDO's transmission type is dictated by the nature of the event that triggered its transmission. There are two configurable transmission modes for PDOs:

- *synchronous objects*—Transmission time is relative to the SYNC message.
- *asynchronous objects*—Transmission time is relative to the message's defined priority.

Triggering Modes

The CANopen communication profile recognizes three modes of message triggering:

- *object-specific event*—A transmission of this type is triggered according to an event specified in the device profile.
- *remote request reception*—Asynchronous PDO transmission can be triggered upon receipt of a remote request from another device.
- *SYNC window expiration*—Reception of the SYNC object can trigger synchronous PDO transmission before the expiration of the SYNC window.

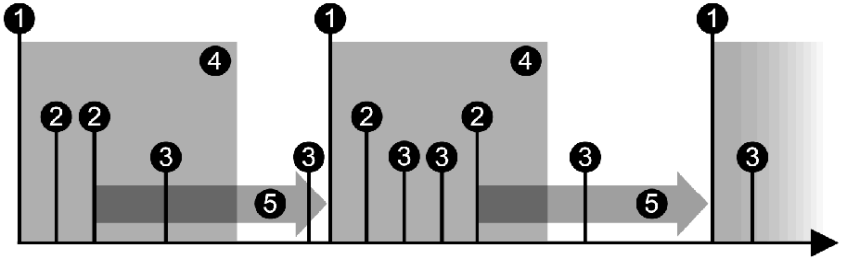
Synchronous Objects

Synchronous PDOs are transmitted within the SYNC window that follows the SYNC object. The interval between SYNC objects is specified by the *communication cycle period* parameter.

The SYNC object and its associated device functionality are represented by three different entries in the object dictionary:

- COB-ID SYNC message (index 1005h)
- communication cycle period
- SYNC window length

The figure below shows the periodic transmission of SYNC messages and synchronous and asynchronous PDOs, relative to the SYNC window:



1 SYNC message
2 synchronous PDO
3 asynchronous PDO
4 SYNC window
5 communication cycle period (the interval between the last synchronous PDO in the window to the next SYNC object)

In general, the synchronous transmission of PDOs guarantees that devices may arrange to sample process variables from a process environment and apply their actuation in a coordinated fashion.

A device that *consumes* SYNC messages will *provide* synchronous PDO messages. The reception of a SYNC message controls the application's interaction with the process environment according to the contents of a synchronous PDO. The synchronous mechanism is intended to transfer commanded values and actual values on a fixed (timely) base.

PDO transmission types are described in the following table.

transmission type	Cyclic	Acyclic	Synchronous	Asynchronous	RTR Only
0		X	X		
1–240	X		X		
241–251	reserved	—	—	—	—
252			X		X
253				X	X
254				X	
255				X	

Synchronous transmission types (0 to 240 and 252) use PDOs that are transmitted relative to the SYNC object. Preferably, devices using the SYNC object to trigger input or output data transmissions will use it in conjunction with the previous RxPDO or TxPDO. Details of this mechanism depend on the device type and are defined in the device profile. Functions for different transmission types are:

- 0—A message of this type is transmitted according to the reception of the SYNC message.
- 1 to 240—These values represent PDOs that are transferred synchronously and cyclically. The transmission type indicates the number of SYNC objects required for triggering PDO transmission or reception.
- 252 to 253—PDOs of this type are sent by remote transmission request only. At transmission type 252, the data is updated (but not sent) immediately after the reception of the SYNC object. At transmission type 253, the data is updated at the reception of the remote transmission request (hardware and software restrictions may apply). These values are only possible for TxPDOs.
- 254—TxPDOs of this type are associated with manufacturer-specific application event (listed in the object dictionary as manufacturer-specific objects).

Cyclic and Acyclic PDOs

Synchronous PDOs are either *cyclic* or *acyclic*. Cyclic PDOs are transmitted upon the reception of some designated number of SYNC objects. For instance, a cyclic PDO may be transmitted after the reception of every third SYNC object. Acyclic PDOs are transmitted after the reception of *every* SYNC object, but only when an internal, designated event (like a change of state) has occurred within the device.

Asynchronous Transmission

Unlike *synchronous* PDOs, an *asynchronous* PDO's transmission is triggered by events not related to the SYNC object, possibly within the device itself. Asynchronous PDO and SDO messages can be transmitted at any time according to their priority. Therefore, asynchronous messages can be transmitted within the SYNC window.

Application events that trigger *asynchronous* PDO transmissions can be device-specific, as described in the device profile, or manufacturer-specific, as described in the manufacturer's documentation.

Default Transmission Mode

For the CANopen NIM, the default transmission mode for default PDOs is asynchronous at an event-driven base (transmission type 255) in accordance with DS-401. This means the PDO will be transmitted on the fieldbus if there is any change of value.

Value changes are determined by the module's configured transmission type on the island bus.

CANopen Emergency Messages

Introduction

Emergency messages are the messages of highest priority on CANopen networks. When a device experiences an internal failure, it transmits an emergency message (available to all network nodes) on the fieldbus.

An emergency message is transmitted only once per error event. If no new errors occur on the device, no additional emergency messages are sent.

Emergency Message Format

The emergency message is always eight bytes. The format is according to the following table:

COB-ID	D1	D2	D3	D4	D5	D6	D7	D8
0x80 + node ID	emergency error code		error register	manufacturer-specific error field				

The first three bytes of the message indicate the error type. When the error disappears, the NIM will report the disappearance on the fieldbus with error code 0000 in the emergency message. (This is called emergency message recovery.) The remaining errors are shown in the error register *(see page 74)*.

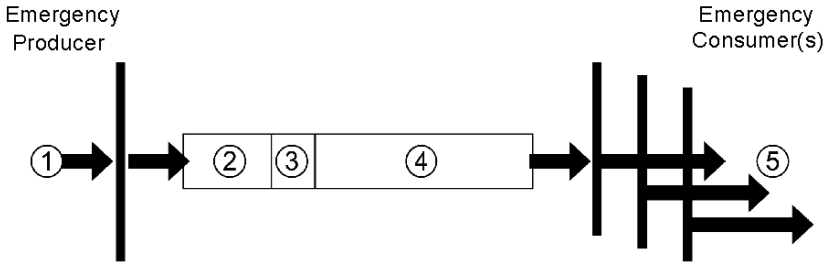
Error registers are discussed in more detail in CANopen Error Detection and Confinement *(see page 102)*.

NOTE: The emergency error code and the error register *(see page 74)* are defined in CANopen DS-301.

The error code is also presented in object 1003 *(see page 74)*.

Error Code	Description
8110h	**CAN overrun (objects lost)**
8120h	**CAN in error passive mode**
8130h	**life guard error or heartbeat error**
8140h	**recovered from bus-off**
8210h	PDO not processed because of length error
FF00	device-specific

The structure of the emergency message is shown in the figure:



1 request
2 emergency error code (2 bytes)
3 error register (1 byte)
4 manufacturer-specific error field (5 bytes)
5 indication(s)

The error register byte is presented in object 1001.

Error Register Bit	Description
0	generic error—set when any error occurs
1	0—not used
2	0—not used
3	0—not used
4	fieldbus communication error—set when: • error status bit is set • node guarding fails • heartbeat fails
5	0—not used
6	0—not used
7	manufacturer-specific error—set when any error (except fieldbus communication error) occurs

Manufacturer-Specific Error Field

The manufacturer-specific error field is optional in CANopen. The CANopen NIM uses these five bytes to provide more information about the error type. The manufacturer-specific error field is structured according to the following table:

Description	Error Code (D4)	Parameter 1 (D5)	Parameter 2 (D6)	Parameter 3 (D7)	Parameter 4 (D8)
island bus fatal error	0x01	island bus state low byte	island bus state high byte	global_bits low byte	global_bits high byte
island bus state exception (configuration mismatch, stopped)	0x02	island bus state low byte	island bus state high byte	global_bits low byte	global_bits high byte
island bus error passive (128 error frames on island bus)	0x03	island bus state low byte	island bus state high byte	global_bits low byte	global_bits high byte
island bus emergency received (from island module)	0x05	island node ID	0x00	0x00	0x00
Advantys configuration software mastery of outputs	0x06	0x00	0x00	0x00	0x00
CANopen fieldbus DLL-error (busoff, overrun, etc.)	0x80	DLL error code	0x00	0x00	0x00
FBH error	0x81	FBH error code	0x00	0x00	0x00
CANopen fieldbus guarding error (lifeguard or heartbeat error	0x82	0x00	0x00	0x00	0x00
CANopen fieldbus short PDO length	0x83	0x00	0x00	0x00	0x00

Error Detection and Confinement for CAN Networks

Introduction

Methodologies that CAN-based networks implement for detecting errors and isolating nodes that produce errors are briefly discussed here.

NOTE: These topics are discussed in greater detail at the CAN in Automation Website (http://www.can-cia.de/).

Error Detection

CAN-based networks use several error detection mechanisms at the bit and message levels.

Two error detection mechanisms are implemented at the bit level:

- *bit monitoring*—After transmitting a message, a CAN node monitors the bit level (in the arbitration field) of the message on the bus. Disagreement between the corresponding bits in the transmitted and monitored messages (because of errors either in the transmitter or on the bus) signals a bit error flag.
- *bit stuffing*—After the transmission of five consecutive identical bits, the transmitter will add (*stuff*) a single bit of opposite polarity to the outgoing bit stream. Receiving nodes will remove (*unstuff*) this extra bit before processing the data. If six identical bits are transmitted consecutively, a stuff error flag is signaled.

Three error detection mechanisms are implemented at the message level:

- *frame check*—CAN-based networks must implement predefined bit values in certain fields of transmitted messages. When the CAN controller detects an invalid value in a bit field, a frame form error is signaled.
- *acknowledgement check*—When a CAN node receives a message, it returns a dominant bit in the message's ACK slot to the transmitter. Otherwise, the transmitter reads the recessive bit in the ACK slot and determines that the message was not received by the intended node(s). An acknowledgement error is signaled.
- *cyclic redundancy check*—Each CAN message has a 15-bit CRC (cyclic redundancy check) that is calculated by the transmitter according to the message's content. The receiving nodes recalculate the CRC field. Disagreement between the two codes indicates a difference between the transmitted message and the one received. In this case, a CRC error flag is signaled.

Error Confinement

The first CAN controller on the bus to detect one of the described errors will transmit the appropriate error flag. Owing to their high priority (only the emergency message is higher), error flags disrupt bus traffic. Other nodes detect the flag (or the original error) and discard the message. CAN's error confinement mechanism distinguishes between temporary errors and permanent failures.

The CAN controller on each node has two dedicated error count registers. Receive errors are accumulated in the *receive error counter* and are given a value of 1. Transmit errors are accumulated in the *transmit error counter* and are given a value of 8. Error-free messages decrement the appropriate (receive or transmit) error registers. The values in the registers dictate the error confinement states of network nodes.

CAN networks define three states in the fault confinement state machine:

- *error active state*—An *error active* node (one operating normally) will transmit error active flags when it detects errors on the bus so that all nodes can abort the offending message. In this state, the error active node assumes it is not the source of the errors.
- *error passive state*—If either error count register exceeds 127, the node enters the *error passive* state. An error passive node transmits error passive flags when it detects errors. These nodes can transmit and receive information, but they may not be able to flag the errors they detect on the fieldbus. Successful operations will decrement the appropriate error registers, eventually returning the node to the *error active* state.
- *bus-off state*—If a node's transmit error counter exceeds 255, the node assumes it is faulty and enters the *bus-off* state. In this way, a repeatedly (or permanently) faulty device will not be active on the bus until the user addresses the issue. Communications between other nodes on the fieldbus will continue as normal.

Application Examples

5

Introduction

This chapter describes how to configure an Advantys STB island on a CANopen network. The described master is a Telemecanique Premium PLC with a TSX CPP 100 CANopen master card. We have used Sycon configuration software (TLX L FBC 10 M) by Hilshcer in the application example.

What's in this Chapter?

This chapter contains the following topics:

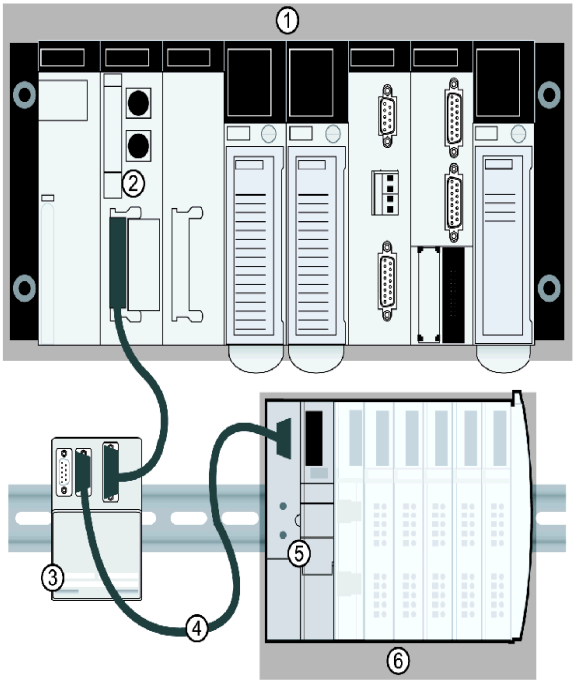
Assembling the Physical Network

Summary

Before describing the CANopen fieldbus master configuration process, take a look at the required hardware connections. The connection figure below shows the components involved in the application example. An assembly procedure is then described.

Connection Diagram

The following diagram shows the connections between a Premium PLC and an STB NCO 2212 NIM over a CANopen network:



1 Premium controller configuration
2 TSX CPP 100 CANopen master PCMCIA card
3 TSX CPP ACC1 CANopen tap junction
4 CANopen network cable (not supplied)
5 STB NCO 2212 CANopen NIM
6 Advantys STB island

Putting the Network Together

The following procedure describes the connections that you need to make to construct a physical CANopen network.

⚠ CAUTION

UNINTENDED EQUIPMENT OPERATION

Read and understand this manual and the Premium user's manual before installing or operating this equipment. Installation, adjustment, repair, and maintenance of this equipment must be performed by qualified personnel.

- Disconnect all power to the Premium PLC before making the network connection.
- Place a DO NOT TURN ON sign on the system power disconnect.
- Lock the disconnect in the open position.

You are responsible for conforming to all applicable code requirements with respect to grounding all equipment.

Failure to follow these instructions can result in injury or equipment damage.

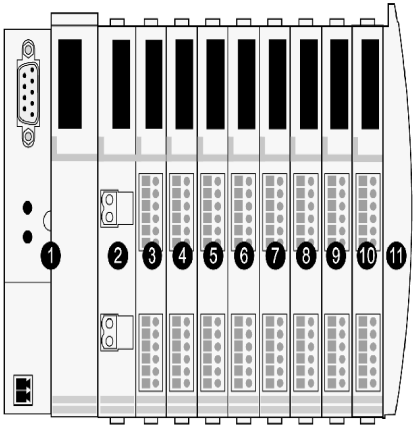
Step	Action
1	Install the TSX CPP 100 CANopen master PCMCIA card in the desired slot on the Premium CPU. (The connection diagram above shows the card in slot 2.)
2	Plug the PCMCIA cable into the TSX CPP ACC1 CANopen tap junction.
3	Using the rotary switches *(see page 26)* on the STB NCO 2212 NIM, set the island to the desired CANopen network node address *(see page 28)*.
4	The CANopen network cable and end connectors (not supplied) should be manufactured in accordance with CiA DRP 303-1.
5	Place the island on the network by connecting the TSX CPP ACC1 CANopen tap junction to the STB NCO 2212 NIM with the CANopen cable.

Sample Island Assembly

The sample I/O system implements a variety of analog and digital modules.

NOTE: The example uses a Telemecanique Premium PLC master device (with a TSX CPP 100 CANopen master card), but the basic configuration of the NIM and the island I/O is master-independent when using the SyCon configuration software.

The following Advantys STB island modules are used in the example:



1 STB NCO 2212, CANopen NIM
2 STB PDT 3100, 24 VDC Power Distribution Module
3 STB DDI 3230, 24 VDC 2-channel digital input module (2 bits of data, 2 bits of status)
4 STB DDO 3200, 24 VDC 2-channel digital output module (2 bits of data, 2 bits of echo output data, 2 bits of status)
5 STB DDI 3420, 24 VDC 4-channel digital input module (4 bits of data, 4 bits of status)
6 STB DDO 3410, 24 VDC 4-channel digital output module (4 bits of data, 4 bits of echo output data, 4 bits of status)
7 STB DDI 3610, 24 VDC 6-channel digital input module (6 bits of data, 6 bits of status)
8 STB DDO 3600, 24 VDC 6-channel digital output module (6 bits of data, 6 bits of echo output data, 6 bits of status)
9 STB AVI 1270, +/-10 VDC 2-channel analog input module (16 bits of data [channel 1], 16 bits of data [channel 2], 8 bits of status [channel 1], 8 bits of status [channel 2])
10 STB AVO 1250, +/-10 VDC 2-channel analog output module (8 bits of status [channel 1], 8 bits of status [channel 2], 16 bits of data [channel 1], 16 bits of data [channel 2])
11 STB XMP 1100 termination plate

The I/O modules in the above island assembly have the following island bus addresses:

I/O Model	Module Type	Island Bus Address
STB DDI 3230	two-channel digital input	1
STB DDO 3200	two-channel digital output	2
STB DDI 3420	four-channel digital input	3
STB DDO 3410	four-channel digital output	4
STB DDI 3610	six-channel digital input	5
STB DDO 3600	six-channel digital output	6
STB AVI 1270	two-channel analog input	7
STB AVO 1250	two-channel analog output	8

The NIM, the PDM, and the termination plate do not consume island bus addresses, and they do not exchange data or status objects with the fieldbus master.

Before You Begin

Before you start configuring the NIM:

- The Advantys STB modules should be assembled and installed.
- The baud *(see page 27)* and node address *(see page 28)* of the CANopen NIM should be set.
- You should have the basic EDS *(see page 64)* file that was supplied with the CANopen NIM.

Data and Status Objects of Advantys STB I/O Modules

Introduction

When configuring PDOs, the size of the data objects and status objects must be known. The status data of the digital I/O and analog I/O is mapped by default to object 6000 *(see page 88)* as digital input data. Therefore, there must already be enough blocks selected in the PDO for this purpose. Care must also be taken to determine the manner in which the PLC will view the data and status objects to facilitate the proper addressing for application use.

NOTE: The discussion in this topic makes reference to the island assembly *(see page 107)* described elsewhere.

Data Objects

Data object sizes for Advantys STB island modules are shown in the following table:

Type of I/O module	Input direction (from island)	Output direction (from PLC)
digital inputs (see 1)	data = < 1 byte (obj. 6000)	—
	status = < 1 byte (obj. 6000) (see 2)	—
digital outputs (see 1)	echo output data = < 1 byte (obj. 6000) (see 2)	data = < 1 byte (object 6200)
	status = < 1 byte (obj. 6000) (see 2)	—
analog inputs, channel 1 (see 3)	data 2 byte (obj. 6401)	—
	status 1 byte (obj. 6000) (see 2 and 4)	—
analog inputs, channel 2 (see 3)	data 2 byte (obj. 6401)	—
	status 1 byte (obj. 6000) (see 2 and 4)	—
analog outputs, channel 1 (see 3)	status 1 byte (obj. 6000) (see 2 and 4)	data 2 byte (object 6411)
	—	—
analog outputs, channel 2 (see 3)	status 1 byte (obj. 6000) (see 2 and 4)	data 2 byte (object 6411)
	—	—

1. Data sizes are based on modules with 8 (or fewer) channels.
2. Not available for every module. Check *The Advantys Hardware Components Reference Guide* (890 USE 172 00) for relevant modules.
3. Data sizes are based on 16-bit resolution.
4. Because this object is mapped by default, you must account for the size of the status data when you initially configure the digital input PDOs in object 6000 *(see page 88)*.

Bit-packing Rules

Bit-packing allows bits associated with the objects for each I/O module to be combined in the same byte whenever possible. The following rules apply:

- Bit-packing follows the addressing order of the island bus I/O modules, from left to right starting with the primary segment.
- The data object (or echo output data object) for a specific module precedes the status object for that module, if available.
- Status objects and data objects for the same or different I/O module may be packed in the same byte, if the size of the combined objects is eight bits or less.
- If the combination of objects requires more than eight bits, the objects will be placed in separate contiguous bytes. A single object can not be split over two byte boundaries.
- By default, data for analog modules is packed in separate PDOs from digital data.
- The status for analog modules (if available) is packed with the digital data.

PLC Data and Status Object View

The table below shows the data for the sample island *(see page 107)* as it will appear in the input and output words of the PLC (in this case, the Telemecanique Premium). The table shows how digital data is bit-packed for optimization, and how data, status, and echo output data (from outputs) appear in the PLC as the same data type (*digital input data*).

The following tables assume the implementation of:

- default island bus mapping (no influence from the Advantys configuration software)
- default CANopen fieldbus mapping (with SyCon)
- default auto-addressing of Premium and SyCon

Also, *N* refers to the island node number in the tables. That is, *N1* represents the first addressable *(see page 48)* node (module) on the sample island *(see page 107)* bus, *N2* the second, and so forth.

PLC data view inputs are shown in the following table:

Word	Byte	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	PDO #
1	1	N2 status		N2 echo output data		N1 status		N1 data		1
	2	N3 status				N3 data				
2	3	N4 status				N4 echo output data				
	4	empty (set to 0)		N5 data						
3	5	empty (set to 0)		N5 status						
	6	empty (set to 0)		N6 echo output data						
4	7	empty (set to 0)		N6 status						
	8	N7 (channel 1) status								

Word	Byte	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	PDO #
5	9	N7 (channel 2) status								2
	10	N8 (channel 1) status								
6	11	N8 (channel 2) status								
	12	empty (set to 0)								
7	13	N7 (channel 1) analog input data (low byte)								3
	14	N7 (channel 1) analog input data (high byte)								
8	15	N7 (channel 2) analog input data (low byte)								
	16	N7 (channel 2) analog input data (high byte)								

PLC data view outputs are shown in the following table.

Word	Byte	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	PDO #
1	1	empty set (set to 0)		N4 output data				N2 output data		1
	2	empty set (set to 0)		N6 output data						
2	3	N8 (channel 1) analog output data (low byte)								2
	4	N8 (channel 1) analog output data (high byte)								
3	5	N8 (channel 2) analog output data (low byte)								
	6	N8 (channel 2) analog output data (high byte)								

Configuring a CANopen Master for Use with the STB NCO 2112 NIM

Summary

These instructions are for configuring the Premium PLC master for use with a CANopen NIM as the head of an Advantys STB island node.

Before You Begin

To use this application example, you should have a working familiarity with both the CANopen fieldbus protocol and the SyCon configuration software.

Before you begin, make sure:

- your Advantys STB modules are fully assembled and installed according to your particular system, application, and network requirements
- you have properly set the baud *(see page 27)* and node address *(see page 28)* of the CANopen NIM
- you have the basic EDS file that was supplied with the STB NCO 2212 CANopen NIM (also available at *www.schneiderautomation.com*)

Importing the NIM's Basic EDS

You need to import the NIM's basic EDS file to the SyCon tool. Without access to the EDS file, the NIM is unavailable for configuration by SyCon. To import the EDS file:

Step	Action
1	Start the SyCon configuration software.
2	From the File *File* menu, select New/CANopen. Click OK.
3	From the File menu, select CopyEDS. Select the directory that contains the NIM's EDS file and, when prompted, accept its corresponding bitmaps.

With the EDS stored in SyCon's database, you can now see *Advantys* in the Nodes pick list.

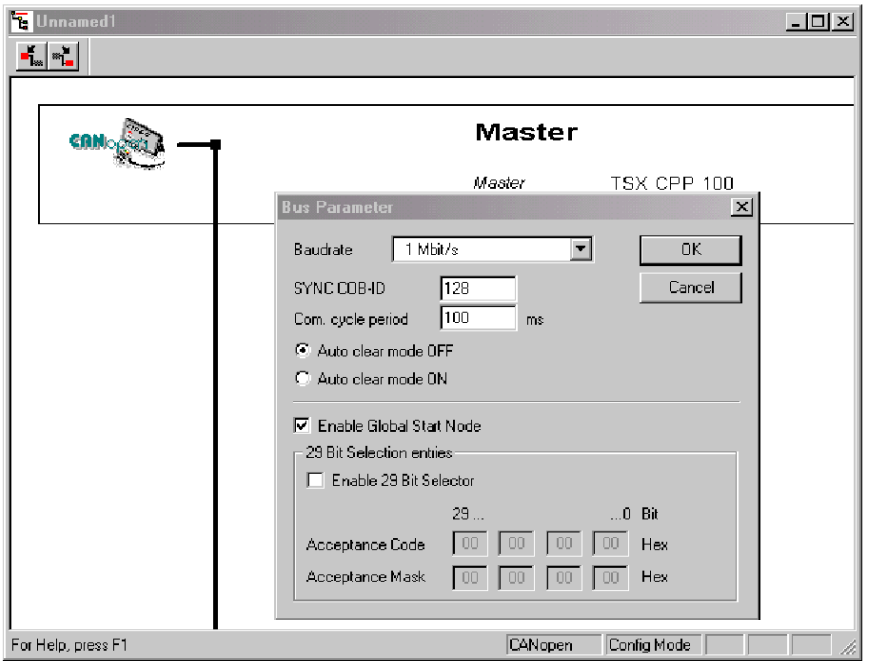
Configuring the Premium PLC

This procedure tells you how to establish the Premium PLC as the master for starting and maintaining the bus:

Step	Action	Comment
1	From the Insert menu, select Master.	
2	From the Insert Master window, select *TSX CPP 100*. Then click Add and OK.	The master will appear in the topology editor screen.
3	From the Settings menu, select Bus Parameters.	Ensure that your configured baud matches the rate previously selected for the NIM.
4	Ensure that the SYNC COB-ID is 128 for the single bus master.	For the example, we will use a single-master network. On a multi-master system, 128 is the COB-ID of the first master.
5	Select the desired Auto clear mode.	Auto clear defines the behavior of the master if communication to a node breaks down or is interrupted.
6	If there is only one master on the bus, check Enable Global Start Node.	As the default Premium setting, Enable Global Start Node is already checked.
7	Click OK and save the file.	The Premium PLC is now the bus master.

The Bus Parameters Dialogue Box

The Bus Parameters dialogue box should resemble the following figure after you've entered parameters according to the above procedure:



About Auto Clear Mode

With Auto clear mode ON selected (checked), the master will stop communication to all active nodes during a communication failure until such time as communication is reestablished or timed out. With Auto clear mode OFF selected, communication failure with a single node does not affect the communication channel to other active nodes. The master will continue trying to restore communications with the faulted node until it is restored or timed out.

Configuring the STB NCO 2212 NIM as a CANopen Network Node

Introduction

These instructions are for configuring an Advantys STB island as a node on a CANopen network using the SyCon configuration software. This requires you to create RxPDOs and TxPDOs that reflect the sum of the possible digital and analog inputs and outputs.

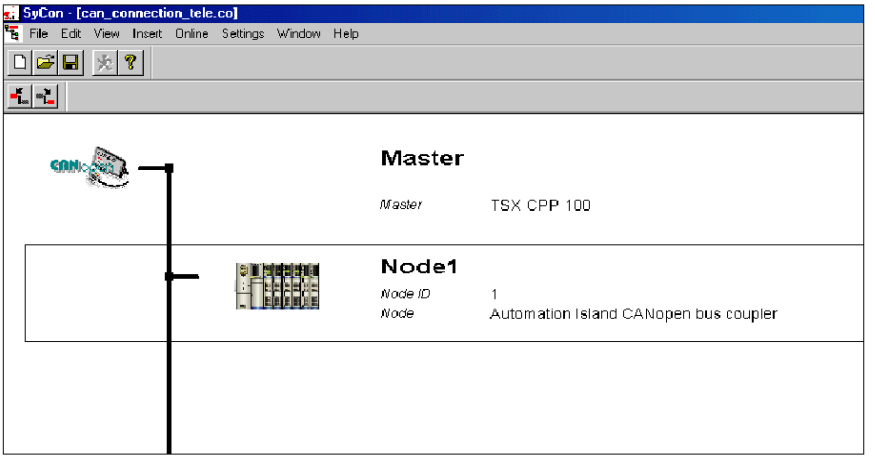
Configuring the Island Node

These instructions are for configuring the CANopen NIM and island modules as a single node on a CANopen network.

Step	Action	Comment
1	From the Insert menu, select Node.	After clicking insert slave, place the node cursor after the master in the topology editor screen *(see page 117)*.
2	In the Insert Node window, set Vendor and Profile to All in the Node Filter area.	
3	Select *Advantys STB CANopen NIM* in the EDS pick list and click the Add tab.	*Advantys STB CANopen NIM* appears in the right window list.
4	Define the node ID or use the default.	You can add a brief description of the node ID, if desired. Do not type spaces in the description.
5	Click OK.	The Advantys icon should appear as a node in the topology editor screen.

The Topology Editor Screen

The topology editor screen should resemble the following figure after you've inserted the CANopen node as a slave using the above procedure:



Defining PDOs

You must now choose specific PDOs for data transmission. Using the sample island assembly *(see page 107)*, you can define and map appropriate PDOs. Then you will pick and map modules for the physical network example.

In this example, we will use default I/O mapping, defining digital inputs first.

Defining Digital Input PDOs

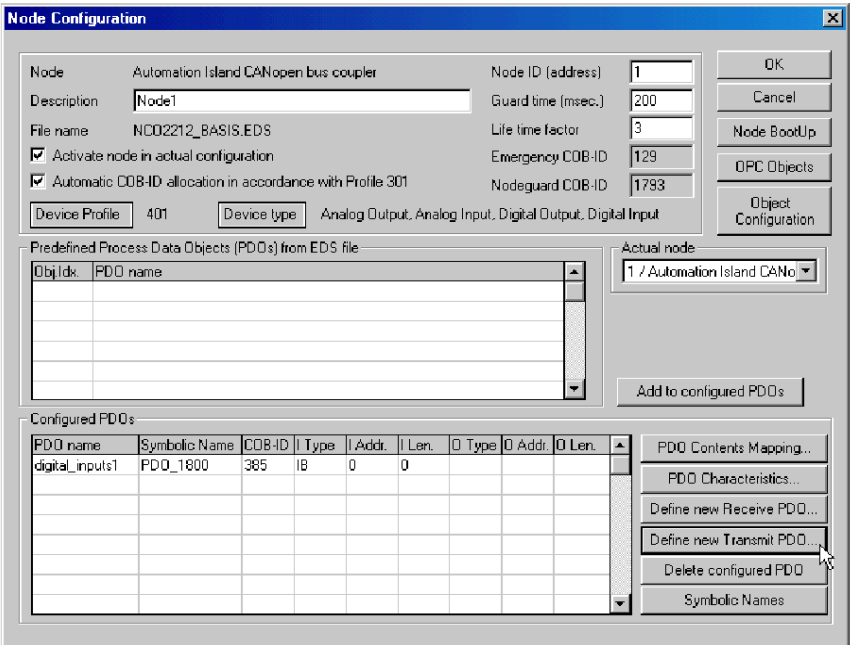
As part of this application example, you will define and map digital input PDOs first. The sample island assembly *(see page 107)* uses three digital input modules, one with two channels, one with four channels, and one with six channels. You must account for 12 bits of input channel data. The remainder of the configuration's 2 bytes of digital input data is allocated for status and feedback data *(see page 110)* from all modules.

Step	Action	Comment
1	In the Node Configuration window *(see page 118)*, click on Define new Transmit PDO. At the prompt, provide a name for this PDO. (Call it *digital_inputs1* for this example.)	The newly named object will appear in the Configured PDOs window.
2	Double-click on the new object in the Configured PDOs window.	The PDO Contents Mapping window appears.

Step	Action	Comment
3	Double-click anywhere in the row for the first object.	The object (at index 6000, subindex 1) will appear in the Mapped Object dictionary window.
4	Double-click anywhere in the row for the first object.	Repeat the above step for all subindexes, 2 through 8, in the Mapped Object dictionary window.
5	Click OK to map the inputs.	You have now mapped 8 bytes of digital input to account for the first 8-byte PDO of possible digital input data.
6	Repeat the steps above, defining a second transmit PDO called *digital_inputs2*.	Your total 2-byte digital input data requires two 8-byte PDOs.

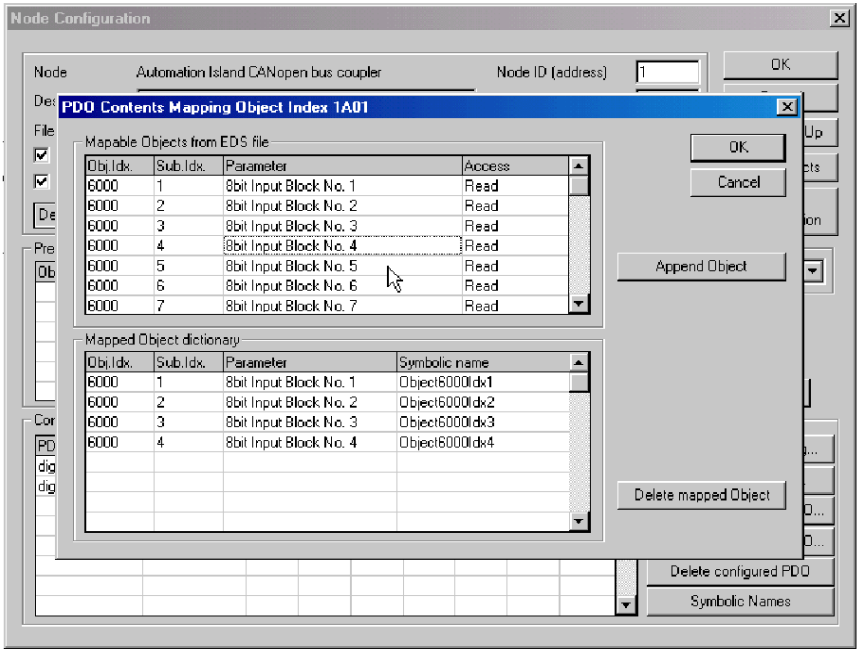
The Node Configuration Window

The following figure shows the Node Configuration window after one TxPDO (for node 1) has been named and mapped:



The PDO Contents Mapping Window

The PDO Contents Mapping window in the figure shows the mapped inputs for the second TxPDO (digital_inputs2).



Defining Digital Output PDOs

You will now define and map digital output PDOs. The sample island assembly *(see page 107)* uses three digital input modules, one with two channels, one with four channels, and one with six channels. Therefore, you must account for all 12 bits of possible digital output data (two blocks of data in one PDO) in your configuration.

Step	Action	Comment
1	In the Node Configuration window *(see page 118)*, click on Define new Receive PDO. At the prompt, provide a name for this PDO. (Call it *digital_outputs1* for this example.)	The newly named object will appear in the Configured PDOs window.
2	Double-click on the new object in the Configured PDOs window.	The PDO Contents Mapping window appears.
3	Double-click anywhere in the row for the second object.	The object (at index 6200, subindex 1) will appear in the Mapped Object dictionary window.

Step	Action	Comment
4	Double-click anywhere in the row for the second object.	The object (at index 6200, subindex 2) will appear in the Mapped Object dictionary window.
5	Click OK to map the outputs.	You have now mapped one PDO with 2 bytes of digital output data.

Defining Analog Input PDOs

Now you will define and map analog input PDOs. The sample island assembly *(see page 107)* uses one two-channel analog input module. You must map one PDO that accounts for both analog input channels.

Step	Action	Comment
1	In the Node Configuration window *(see page 118)*, click on Define new Transmit PDO. At the prompt, provide a name for this PDO. (Call it *analog_inputs* for this example.)	The newly named object will appear in the Configured PDOs window.
2	Double-click on the new object in the Configured PDOs window.	The PDO Contents Mapping window appears.
3	Scroll to the object (index 6401, subindex 1) and double-click anywhere in its row.	The object will appear in the Mapped Object dictionary window. You now need to map an object for the other analog input channel to complete the PDO.
4	Scroll to the object (index 6401, subindex 2) and double-click anywhere in its row.	The object will appear in the Mapped Object dictionary window.
5	Click OK to map the inputs.	You have now mapped one PDO that accounts for 2 channels of possible analog input data.

Defining Analog Output PDOs

Now you will define and map analog output PDOs. The sample island assembly *(see page 107)* uses one two-channel analog output module. You must map one PDO that accounts for both analog output channels.

Step	Action	Comment
1	In the Node Configuration window *(see page 118)*, click on Define new Receive PDO At the prompt, provide a name for this PDO. (Call it *analog_outputs* for this example.)	The newly named object will appear in the Configured PDOs window.
2	Double-click on the new object in the Configured PDOs window.	The PDO Contents Mapping window appears.
3	Scroll to the object (index 6411, subindex 1) and double-click anywhere in its row.	The object will appear in the Mapped Object dictionary window. You need to continue to map an object for the other analog output channel.

Step	Action	Comment
4	Double-click on the new object in the Configured PDOs window.	The PDO Contents Mapping window appears.
5	Scroll to the object (index 6411, subindex 2) and double-click anywhere in its row.	The object will appear in the Mapped Object dictionary window.
6	Click OK to map the inputs.	You have now mapped one PDO that accounts for 2 channels of possible analog output data.

Defining Transmission Types

You need to define a transmission type (operating mode) for each PDO in your configuration. There are several transmission types and triggering modes available in the *PDO Characteristics* window. For digital inputs and outputs we will use the default types for this example. View the default types by selecting a PDO from the list of configured PDOs and clicking on the *PDO Characteristics* tab.

Synchronous PDOs are those in which the transmission is related to the SYNC message that the master sends cyclically. An asynchronous PDO is one in which transmission is not related to the SYNC message; transmission is dictated by the message's priority.

The values listed as Resulting CANopen-specific transmission types (in the PDO Characteristics window) are:

- 0—This message will be transmitted synchronously, with respect to the SYNC message.
- 1 to 240—A PDO of this type is transmitted synchronously and cyclically. The value indicates the number of SYNC messages between two transmissions of the PDO.
- 252 to 253—A PDO of this type is associated with an event with no immediate notification. This PDO is only transmitted upon the reception of a remote transmission request.
- 252—This data is updated immediately after the reception of the SYNC message, but it is not sent.
- 253—PDO data is updated upon the reception of a remote transmission request.
- 254—The PDO is associated with a manufacturer-specific application event.

These values are automatically assigned when selecting the appropriate transmission and trigger modes. To view these parameters, select a PDO from the list of configured PDOs and click on the PDO Characteristics tab to view the object's transmission and triggering modes.

Saving the CANopen Configuration

Summary

Saving your configuration ensures that your changes will be stored in the NIM's Flash memory. Otherwise, the object's default settings will be implemented at the next power cycle.

Setting Object 1010

If you changed any of the default values in your node configuration, it will be necessary to set object 1010 to sub index 1 (save all parameters).

Step	Action	Comment
1	From the Node Configuration screen, click on the Object Configuration button.	The Object Configuration window opens.
2	From the Object Configuration window, scroll to object 1010 and double-click anywhere in the row.	Object 1010 will appear in the Configured Objects window.
3	From the predefined supported objects screen, double-click on object 1010, subindex 1 (*save all parameters*).	It should appear in the configured objects window.
4	Enter *00* in the chosen value line of the configured objects window.	The *00* value is for the example only.
5	Click OK to save the changes.	

Saving the Configuration

Saving the configuration from this point is similar to any computer application. After startup, you can reference and use the I/O data you have configured in your CANopen system.

Step	Action	Comment
1	From the *File* menu, select *Save*.	The *Save As* dialogue box appears.
2	Give the configuration a unique name and direct it to the folder of your choice.	You may want to save the configuration (.co) file to the ***PL7 user*** directory in which the Premium PLC resides.
3	Click *Save*.	The configuration is written to the NIM's flash memory during the next startup sequence.

Configuring CANopen NIMs for use with Hi-Density I/O Modules

16-bit Considerations - Digital I/O

Auto-configuration of an Advantys STB island, which includes one or more 16-bit digital I/O modules and a CANopen NIM, will not automatically map all I/O data registers to a PDO. In order to map the 16-bit digital I/O data to a PDO, you must use a CANopen configuration tool.

For example, suppose that your Advantys STB island consists of a CANopen NIM, an STB PDT 3100 power distribution module, an STB DDI 3725 16-bit digital input module, and an STB DDO 3705 16-bit digital output module. At start-up, the auto-configuration process will not map inputs or outputs to any PDOs in the NIM. Instead, you must manually map this data.

The 16-bit input data from the STB DDI 3725 resides in Object Dictionary (OD) index 6100h, sub-index 01h. The 16-bit output data of the STB DDO 3705 is in OD index 6300h, sub-index 01h. If you want to map all of these values to PDO 1, for example, you must connect your CANopen configuration tool to the NIM, start it up, and then write the following mapping values to the NIM's OD using the CANopen configuration tool according to its directions:

Receive PDO 1 mapping:

- Index 1600h, sub-index 0 = 1
- Index 1600h, sub-index 1 = 6300 01 10

Transmit PDO 1 mapping:

- Index 1A00h, sub-index 0 = 1
- Index 1A00h, sub-index 1 = 6100 01 10

16-bit Considerations - Analog I/O (STB ACI 1320, STB ACI 8320, STB ACO 0220)

Auto-configuration of an Advantys STB island, which includes one or more of these analog modules and a CANopen NIM, will not automatically map all I/O data registers to a PDO. In order to map the 1analog input data to a PDO, you must use a CANopen configuration tool.

For example, suppose that your Advantys STB island consists of a CANopen NIM, an STB PDT 3100 power distribution module, an STB ACI 0320 or an STB ACI 8320 analog input module, and an STB ACO 0220 analog output module. At start-up, the auto-configuration process will not map inputs or outputs to any PDOs in the NIM. Instead, you must manually map this data.

The 16-bit analog input data from the STB ACI 0320 and the STB ACI 8320 resides in Object Dictionary (OD) starting at index 2200h. The 16-bit analog output data of the STB ACO 0320 resides in OD starting at index 3200h. If you want to map all of these values to a PDO, for example, you must connect your CANopen configuration tool to the NIM, start it up, and then write the displayed mapping values to the NIM's OD using the CANopen configuration tool according to its directions.

Advanced Configuration Features

6

Introduction

This chapter describes the advanced and/or optional configuration features that you can add to an Advantys STB island.

What's in this Chapter?

This chapter contains the following topics:

STB NCO 2212 Configurable Parameters

Functional Characteristics

This topic discusses the configuration of CANopen NIM parameters using the Advantys configuration software.

The following operating parameters are user-configurable:

- data size (in words) of PLC output data transmitted to the HMI panel and HMI input data sent to the PLC
- maximum node ID for the last CANopen device
- enabling/disabling the remote virtual placeholder option *(see page 173)* using the fieldbus handler control word

General Information

To get general information about the NIM module (model name, version number, vendor code, etc.):

Step	Action	Comment
1	Open your configuration with the Advantys configuration software.	The NIM is the leftmost module in your island assembly.
2	Double-click on the NIM in the configuration workspace.	The *module editor* window appears.
3	Select the *General* tab.	The *General* tab gives general information about the NIM.

Accessing the NIM Parameters List

To access the NIM values that are configurable:

Step	Action	Comment
1	Open the *module editor*.	
2	Select the *Parameters* tab.	Configurable parameters are on this tab.
3	Expand the *NIM Parameters List* by clicking on the plus (+) sign.	The configurable NIM parameters become visible.

Reserved Sizes (HMI to PLC)

The network interprets data from the HMI as input and reads it from the input data table in the process image. This table is shared with data from all input modules on the island bus. When the reserved size (HMI to PLC) value is selected, the range of available data sizes (in words) appears in the window (see the above figure). The maximum size includes both the input data produced by the island modules and the HMI to PLC data. Therefore, space that you reserve for the HMI to PLC data—plus the input data from the island bus modules—must not exceed the maximum value shown. For example, if your input modules produce eight words of input data, you can reserve only the remaining 112 words (out of 120 maximum) of the input data table for the HMI to PLC data.

Reserved Sizes (PLC to HMI)

The network transmits data to the HMI as output by writing it to the output data table in the process image. This table is shared with data for all output modules on the island bus. When the reserved size (PLC to HMI) value is selected, the range of available data sizes (in words) appears in the window (see the above figure). The maximum size includes both the data sent to the island modules and the PLC to HMI data. Therefore, space that you reserve for PLC to HMI data—plus the output data for the island bus modules—must not exceed the maximum value. For example, if your output modules consume three words of output data, you can reserve only the remaining 117 words (out of 120 maximum) of the output data table for the PLC to HMI data.

Reserving Data Sizes

To transfer data to the PLC from a Modbus HMI you must reserve sizes for that data. To reserve these data sizes:

Step	Action	Result
1	In the *module editor*, access the *NIM Parameter List*.	
2	Double-click in the *Configured Value* column next to the *Reserved Size (Words) of HMI to PLC table*.	The value is highlighted.
3	Enter a value that represents the data size that will be reserved for data sent from the HMI panel to the PLC.	The value you enter *plus* the data size of your island can not exceed the maximum value. If you accept the default (0), no space will be reserved in the HMI table in the process image.

Step	Action	Result
4	Repeat the above steps to select a value for the *Reserved Size (Words) of PLC to HMI table* row.	
5	Press *OK* when you have entered the desired data sizes.	

The Fieldbus Handler Control Word

To enable a remote virtual placeholder option *(see page 173)* on the island:

Step	Action	Result
1	In the *module editor*, access the *NIM Parameter List*.	
2	Expand the *Fieldbus Handler Control Word* parameter by clicking on the plus (+) sign.	The Remote Virtual Placeholders parameter becomes visible.
3	Click the drop-down list in the *Configured Value* of the *Remote Virtual Placeholders* parameter. Select a value of 1 to enable the remote virtual placeholder option on the island.	The default value is 0, which disables the remote virtual place holder option.
4	Press *OK*	When the remote virtual placeholder option is enabled, any standard virtual placeholder settings on the individual I/O modules on the island are ignored.

CANopen Device Node IDs

On the Parameters tab, you can set the maximum node ID of the last module on the island bus. Standard CANopen devices follow the last segment of STB I/O modules. CANopen modules are addressed by counting backwards from the value you enter here. The ideal node ID sequence is sequential.

For example, if you have an island with five STB I/O modules and three CANopen devices, a maximum node ID of at least 8 (5 + 3) is required. This will result in node IDs of 1 through 5 for STB I/O modules and 6 through 8 for standard CANopen devices. Using the default ID of 32 (the maximum number of modules the island can support) will result in node IDs of 1 through 5 for STB I/O modules and 30 through 32 for standard CANopen devices. Those unnecessarily high addresses are not desirable if any of your standard CANopen devices have a limited address range.

Assigning the Max. Node ID (CANopen Devices)

To enter the highest node ID used by a CANopen device on the island bus:

Step	Action	Comment
1	In the *module editor*, select the *Parameters* tab.	Configurable parameters are on this tab.
2	In the box next to *Max. node ID on the CANopen extension*, enter a node ID.	This node ID represents the last CANopen module on the island bus.

Configuring Mandatory Modules

Summary

As part of a custom configuration, you can assign *mandatory* status to any I/O module or preferred device on an island. The mandatory designation indicates that you consider the module or device critical to your application. If the NIM does not detect a healthy mandatory module at its assigned address during normal operations, the NIM stops the entire island.

NOTE: The Advantys Configuration Software is required if you want to designate an I/O module or a preferred device as a mandatory module.

Specifying Mandatory Modules

By default, the Advantys STB I/O modules are in a non-mandatory (*standard*) state. Mandatory status is enabled by clicking on the mandatory checkbox on a module or preferred device's **Options** tab. Depending on your application, any number of modules that your island supports can be designated as mandatory modules.

Effects on Island Bus Operations

The following table describes the conditions under which mandatory modules affect island bus operations and the NIM's response:

Condition	Response
A mandatory module is not operating during normal island bus operations.	The NIM stops the island bus. The island enters fallback mode *(see page 137)*. I/O modules and preferred devices assume their fallback values.
You attempt to hot swap a mandatory module.	The NIM stops the island bus. The island enters fallback mode. I/O modules and preferred devices assume their fallback values.
You are hot swapping a standard I/O module that resides to the left of a mandatory module on the island bus, and the island loses power.	When power is restored, the NIM attempts to address the island modules but must stop at the empty slot where the standard module used to reside. Because the NIM is now unable to address the mandatory module, it generates a mandatory mismatch condition. The island does not start when this condition is present.

Recovering from a Mandatory Stop

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION/LOSS OF CONFIGURATION—RST BUTTON WHILE RECOVERING FROM MANDATORY STOP

Pushing the RST button *(see page 58)* causes the island bus to reconfigure itself with factory-default operating parameters, which do not support mandatory I/O status.

- Do not attempt to restart the island by pushing the RST button.
- If a module is unhealthy, replace it with the same module type.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Pushing the RST button *(see page 58)* while recovering from a mandatory stop loads the island's default configuration data.

Hot Swapping a Mandatory Module

If the NIM has stopped island bus operations because it cannot detect a healthy mandatory module, you can recover island bus operations by installing a healthy module of the same type. The NIM automatically configures the replacement module to match the removed module. Assuming that other modules and devices on the island bus are correctly configured and conform to their configuration data as written to Flash memory, the NIM starts of restarts normal island bus operations.

Prioritizing a Module

Summary

Using the Advantys configuration software, you can assign priority to digital input modules in your island assembly. Prioritization is a method of fine tuning the NIM's I/O scan of the island bus. The NIM will scan modules with priority more frequently than other island modules.

Limitations

You can prioritize only modules with digital inputs. You cannot prioritize output modules or analog modules. You can prioritize only 10 modules for a given island.

What Is a Reflex Action?

Summary

Reflex actions are small routines that perform dedicated logical functions directly on the Advantys island bus. They allow output modules on the island to act on data and drive field actuators directly, without requiring the intervention of the fieldbus master.

A typical reflex action comprises one or two function blocks that perform:

- Boolean AND or exclusive-OR operations
- comparisons of an analog input value to user-specified threshold values
- up- or down-counter operations
- timer operations
- the triggering of a latch to hold a digital value high or low
- the triggering of a latch to hold an analog value at a specific value

The island bus optimizes reflex response time by assigning the highest transmission priority to its reflex actions. Reflex actions take some of the processing workload off the fieldbus master, and they offer a faster, more efficient use of system bandwidth.

How Reflex Actions Behave

⚠ WARNING

UNEXPECTED OUTPUT OPERATION

For outputs that are configured to respond to reflex actions, the output state represented in the island's network interface module (NIM) may not represent the actual states of the outputs.

- Turn off field power before you service any equipment connected to the island.
- For digital outputs, view the echo register for the module in the process image to see the actual output state.
- For analog outputs, there is no echo register in the process image. To view an actual analog output value, connect the analog output channel to an analog input channel.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Reflex actions are designed to control outputs independently of the fieldbus master controller. They may continue to turn outputs on and off even when power is removed from the fieldbus master. Use prudent design practices when you use reflex actions in your application.

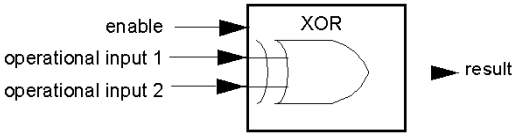
Configuring a Reflex Action

Each block in a reflex action must be configured using the Advantys configuration software.

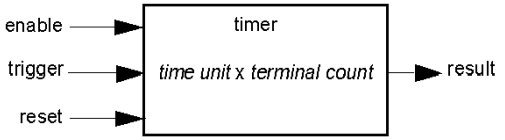
Each block must be assigned a set of inputs and a result. Some blocks also require that you specify one or more user-preset values—a compare block, for example, requires that you preset threshold values and a delta value for hysteresis.

Inputs to a Reflex Action

The inputs to a reflex block include an enable input and one or more operational inputs. The inputs may be constants or they may come from other I/O modules on the island, from virtual modules or outputs from another reflex block. For example, an XOR block requires three inputs—the enable and two digital inputs that contain the Boolean values to be XORed:



Some blocks, such as the timers, require reset and/or trigger inputs to control the reflex action. The following example shows a timer block with three inputs:



The trigger input starts the timer at 0 and accumulates *time units* of 1, 10, 100 or 1000 ms for a specified number of counts. The reset input causes the timer accumulator to be reset.

An input to a block may be a Boolean value, a word value, or a constant, depending on the type of reflex action it is performing. The enable input is either a Boolean or a constant *always enabled* value. The operational input to a block such as a digital latch must always be a Boolean, whereas the operational input to an analog latch must always be a 16-bit word.

You will need to configure a source for the block's input values. An input value may come from an I/O module on the island or from the fieldbus master via a virtual module in the NIM.

NOTE: All inputs to a reflex block are sent on a change-of-state basis. After a change-of-state event has occurred, the system imposes a 10 ms delay before it accepts another change of state (input update). This feature is provided to minimize jitter in the system.

Result of a Reflex Block

Depending on the type of reflex block that you use, it will output either a Boolean or a word as its result. Generally, the result is mapped to an *action module*, as shown in the following table:

Reflex Action	Result	Action Module Type
Boolean logic	Boolean value	digital output
integer compare	Boolean value	digital output
counter	16-bit word	first block in a nested reflex action
timer	Boolean value	digital output
digital latch	Boolean value	digital output
analog latch	16-bit word	analog output

The result from a block is usually mapped to an individual channel on an output module. Depending on the type of result that the block produces, this action module may be an analog channel or a digital channel.

When the result is mapped to a digital or analog output channel, that channel becomes dedicated to the reflex action and can no longer use data from the fieldbus master to update its field device.

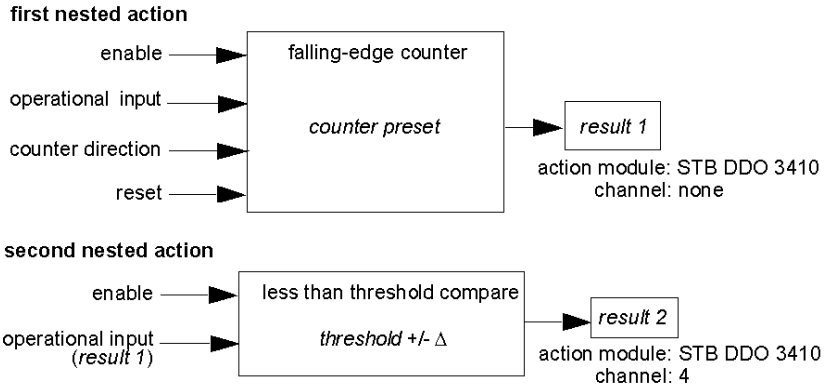
The exception is when a reflex block is the first of two actions in a nested reflex action.

Nesting

The Advantys configuration software allows you to create nested reflex actions. One level of nesting is supported—i.e., two reflex blocks, where the result of the first block is an operational input to the second block.

When you nest a pair of blocks, you need to map the results of both to the same action module. Choose the action module type that is appropriate for the result of the second block. This may mean that in some cases you will need to choose an action module for the first result that does not seem to be appropriate according to the table above.

For example, say you want to combine a counter block and a compare block in a nested reflex action. You want the result of the counter to be the operational input to the compare block. The compare block will then produce a Boolean as its result:



Result 2 (from the compare block) is the result that the nested reflex action will send to an actual output. Because the result of a compare block needs to be mapped to a digital action module, *result 2* is mapped to channel 4 on an STB DDO 3410 digital output module.

Result 1 is used only inside the module—it provides the 16-bit operational input to the compare block. It is mapped to the same STB DDO 3410 digital output module that is the action module for the compare block.

Instead of specifying a physical channel on the action module for *result 1*, the channel is set to *none*. In effect, you are sending *result 1* to an internal reflex buffer where it is stored temporarily until it is used as the operational input to the second block. You are not really sending an analog value to a digital output channel.

Number of Reflex Blocks on an Island

An island can support up to 10 reflex blocks. A nested reflex action consumes two blocks.

An individual output module can support up to two reflex blocks. Supporting more than one block requires that you manage your processing resources efficiently. If you are not careful with your resources, you may be able to support only one block on an action module.

Processing resources are consumed quickly when a reflex block receives its inputs from multiple sources (different I/O modules on the island and/or virtual modules in the NIM). The best way to preserve processing resources is to:

- use the *always enabled* constant as the enable input whenever possible
- use the same module to send multiple inputs to a block whenever possible

Island Fallback Scenarios

Introduction

In the event of a communications interruption on the island or between the island and the fieldbus, output data is put into a fallback state. In this state, output data is replaced with pre-configured fallback values. This makes known the module's output data values when the system recovers from this condition.

Fallback Scenarios

There are several scenarios in which Advantys STB output modules go into their fallback states:

- loss of fieldbus communications: Communications with the PLC are lost.
- loss of island bus communications: There is an internal island bus communications interruption, indicated by a missing heartbeat message from either the NIM or a module.
- change of operating state: The NIM may command the island I/O modules to switch from a running to a non-running (stopped or reset) state.
- missing or non-operating mandatory module: The NIM detects this condition for a mandatory island module.

NOTE: If a mandatory (or any other) module is not operating, it needs to be replaced. The module itself does not go into its fallback state.

In all of these fallback scenarios, the NIM disables the heartbeat message.

Heartbeat Message

The Advantys STB system relies on a heartbeat message to verify the integrity and continuity of communications between the NIM and the island modules. The health of island modules and the overall integrity of the Advantys STB system are monitored through the transmission and reception of these periodic island bus messages.

Because island I/O modules are configured to monitor the NIM's heartbeat message, output modules go into their fallback states if they do not receive a heartbeat message from the NIM within the defined interval.

Fallback States for Reflex Functions

Only an output module channel to which the result of a reflex action *(see page 133)* has been mapped can operate in the absence of the NIM's heartbeat message.

When modules that provide input for reflex functionality are not operating or are removed from the island, the channels that hold the result of those reflex actions go into their fallback states.

In most cases, an output module that has one of its channels dedicated to a reflex action goes to its configured fallback state if the module loses communication with the fieldbus master. The only exception is a two-channel digital output module that has both of its channels dedicated to reflex actions. In this case, the module may continue to solve logic after a loss of fieldbus communication. For more information about reflex actions, refer to the *Reflex Actions Reference Guide*.

Configured Fallback

To define a customized fallback strategy for individual modules, you are required to use the Advantys Configuration Software. Configuration is done channel by channel. You can configure a single module's multiple channels with different fallback parameters. Configured fallback parameters (implemented only during a communications interruption) are part of the configuration file stored in the NIM's non-volatile Flash memory.

Fallback Parameters

You can select either of two fallback modes when configuring output channels with the Advantys Configuration Software:

- *hold last value*: In this mode, outputs retain the last values they were assigned before the fallback condition was triggered.
- *predefined value*: In this (default) mode, you can select either of two fallback values:
  - 0 (default)
  - some value in acceptable range

The permissible values for fallback parameters in the *predefined value* mode for discrete and analog modules and reflex functions appear in the following table:

Module Type	Fallback Parameter Values
discrete	0/off (default)
	1/on
analog	0 (default)
	not 0 (in range of acceptable analog values)

NOTE: In an auto-configured system, default fallback parameters and values are always used.

Saving Configuration Data

Introduction

The Advantys configuration software allows you to save configuration data created or modified with this software to the NIM's Flash memory and/or to the removable memory card *(see page 52)*. Subsequently, this data can be read from Flash memory and used to configure your physical island.

NOTE: If your configuration data is too large, you will receive a message when you attempt to save it.

How to Save a Configuration

The following procedure describes the steps you use to save a configuration data file to Flash memory directly and to a removable memory card. For more detailed procedural information, use the configuration software's online help feature:

Step	Action	Comment
1	Connect the device running the Advantys Configuration Software to the CFG port *(see page 35)* on the NIM.	For NIM modules that support Ethernet communications, you can connect the device directly to the Ethernet port.
2	Launch the configuration software.	
3	Download the configuration data that you want to save from the configuration software to the NIM.	A successful download saves the configuration data to the NIM's flash memory.
4	Install the card *(see page 53)* in the host NIM, then use the **Store to SIM card** command.	Saving the configuration data to the removable memory card is optional. This operation overwrites old data on the SIM card.

Write-Protecting Configuration Data

Introduction

As part of a custom configuration, you can password-protect an Advantys STB island. Only authorized persons have write privileges to the configuration data currently stored in Flash memory:

- Use the Advantys Configuration Software to password-protect an island's configuration.
- For some modules, it is possible to password-protect the island configuration through an embedded web site.

The island runs normally in protected mode. All users have the ability to monitor (read) the activity on the island bus. If a configuration is write-protected, access to it is restricted in the following ways:

- An unauthorized user is unable to overwrite the current configuration data in Flash memory.
- The RST button *(see page 58)* is disabled, and pushing it has no effect on island bus operations.
- The presence of a removable memory card *(see page 52)* is ignored. The configuration data currently stored in Flash cannot be overwritten by data on the card.

NOTE: The STB NIP 2311 NIM never ignores the removable memory card.

Password Characteristics

A password must meet the following criteria:

- It must be between 0 and 6 characters in length.
- Only alphanumeric ASCII characters are permitted.
- The password is case-sensitive.

If password protection is enabled, your password is saved to Flash memory (or to a removable memory card) when you save the configuration data.

NOTE: A password-protected configuration is inaccessible to anyone who does not know the password. Your system administrator is responsible for keeping track of the password and the list of authorized users. If the assigned password is lost or forgotten, you are unable to change the island's configuration.

If the password is lost and you need to reconfigure the island, you need to perform a destructive reflash of the NIM. This procedure is described on the Advantys STB product Web site at *www.schneiderautomation.com*.

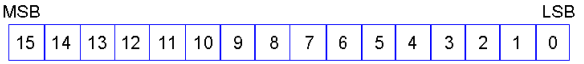
A Modbus View of the Island's Data Image

Summary

A block of Modbus registers is reserved in the NIM to hold and maintain the island's data image. Overall, the data image holds 9999 registers. The registers are divided into contiguous groups (or blocks), each dedicated to a specific purpose.

Modbus Registers and Their Bit Structure

Registers are16-bit constructs. The most significant bit (MSB) is bit 15, which is displayed as the leftmost bit in the register. The least significant bit (LSB) is bit 0, displayed as the rightmost bit in the register:

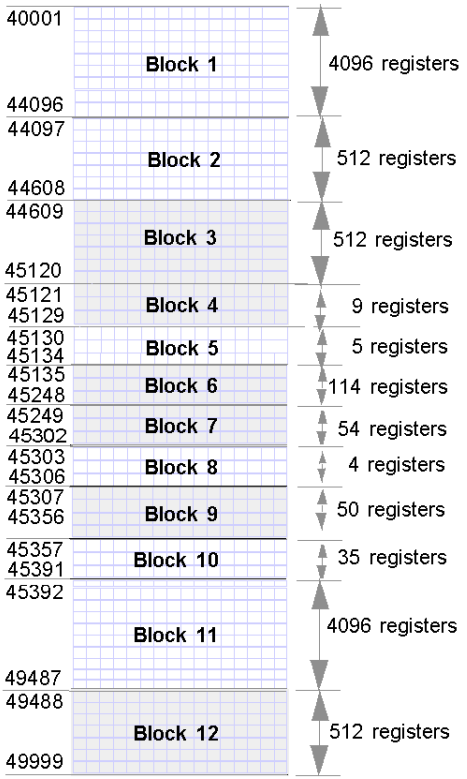


The bits can be used to display operating data or device/system status.

Each register has a unique reference number, starting at 40001. The content of each register, represented by its 0/1 bit pattern, may be dynamic, but the register reference and its assignment in the control logic program remain constant.

The Data Image

The 9999 contiguous registers in the Modbus data image start at register 40001. This figure shows the subdivision of data into sequential blocks:



Block 1 output data process image (4096 registers available)
Block 2 fieldbus master-to-HMI output table (512 registers available)
Block 3 reserved (512 registers available)
Block 4 9-register block reserved for future read/write use
Block 5 5-register RTP Request Block
Block 6 114-register block reserved for future read/write use
Block 7 54-register block reserved for future read/write use
Block 8 4-register RTP Response Block
Block 9 50-register block reserved for future read-only use
Block 10 35 predefined island bus status registers
Block 11 input data/status process image (4096 registers available)
Block 12 HMI-to-fieldbus master input table (512 registers available)

Each block has a fixed number of registers reserved for its use. Whether or not all the registers reserved for that block are used in an application, the number of registers allocated to that block remains constant. This permits you to know at all times where to begin looking for the type of data of interest to you.

For example, to monitor the status of the I/O modules in the process image, look at the data in block 11 beginning at register 45392.

Reading Register Data

All the registers in the data image can be read by an HMI panel connected to the island at the NIM's CFG port *(see page 35)*. The Advantys configuration software reads all this data, and displays blocks 1, 2, 5, 8, 10, 11, and 12 in the Modbus Image screen in its I/O Image Overview.

Writing Register Data

Some registers, usually configured number of registers in block 12 (registers 49488 through 49999) of the data image, may be written to by an HMI panel *(see page 162)*.

The Advantys configuration software or an HMI panel may also be used to write data to the registers in block 1 (registers 40001 through 44096). The configuration software or the HMI panel must be the island bus master in order for it to write to the data image—i.e., the island must be in *test* mode.

The Island's Process Image Blocks

Summary

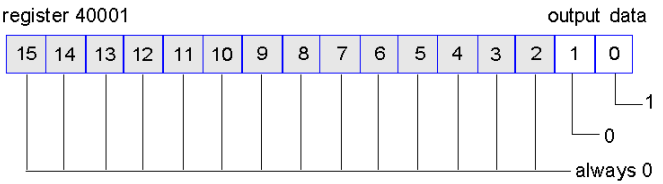
Two blocks of registers in the island's data image *(see page 142)* are the focus for this discussion. The first block is the output data process image, which starts at register 40001 and goes to register 44096. The other block is the input data and I/O status process image, which also consumes 4096 registers (45392 through 49487). The registers in each block are used to report island bus device status and to dynamically exchange input or output data between the fieldbus master and the island's I/O modules.

Output Data Process Image

The output data block (registers 40001 through 44096) handles the output data process image. This process image is a Modbus representation of the control data that has just been written from the fieldbus master to the NIM. Only data for the island's output modules is written to this block.

Output data is organized in 16-bit register format. One or more registers are dedicated to the data for each output module on the island bus.

For example, say you are using a two-channel digital output module as the first output module on your island bus. Output 1 is on and output 2 is off. This information would be reported in the first register in the output data process image, and it would look like this:



where:

- Normally, a value of 1 in bit 0 indicates that output 1 is on.
- Normally, a value of 0 in bit 1 indicates that output 2 is off.
- The remaining bits in the register are not used.

Some output modules, such as the one in the example above, utilize a single data register. Others may require multiple registers. An analog output module, for example, would use separate registers to represent the values for each channel, and might use the 11 or 12 most significant bits to display analog values in IEC format.

Registers are allocated to output modules in the output data block according to their addresses on the island bus. Register 40001 always contains the data for the first output module on the island (the output module closest to the NIM).

Output Data Read/Write Capabilities

The registers in the output data process image are read/write-capable.

You can read (i.e., monitor) the process image using an HMI panel or the Advantys Configuration Software. The data content that you see when you monitor the output data image registers is updated in near-real time.

The island's fieldbus master also writes updated control data to the output data process image.

Input Data and I/O Status Process Image

The input data and I/O status block (registers 45392 through 49487) handles the input data and I/O status process image. Every I/O module on the island bus has information that needs to be stored in this block.

- Each digital input module reports data (the on/off status of its input channels) in one register of input data and I/O status block, then reports its status in the next register.
- Each analog input module uses four registers in the input data and I/O status block. It represents the analog data for each channel in separate registers and the status of each channel in separate registers. Analog data is usually represented with 11- or 12-bit resolution in the IEC format; status in an analog input channel is usually represented by a series of status bits that report the presence or absence of an out-of-range value in a channel.
- Each digital output module reports an echo of its output data to a register in the input data and I/O status block. Echo output data registers are essentially copies of the register values that appear in the output data process image. This data is usually not of much interest, but it can be useful in the event that a digital output channel has been configured for a reflex action. In this case, the fieldbus master can see the bit value in the echo output data register even though the output channel is being updated inside the island bus.
- Each analog output module uses two registers in the input data and I/O status block to report status. Status in an analog output channel is usually represented by a series of status bits that report the presence or absence of an out-of-range value in a channel. Analog output modules do not report data in this block.

A detailed view of how the registers in the input data and I/O status block are implemented is shown in the process image example.

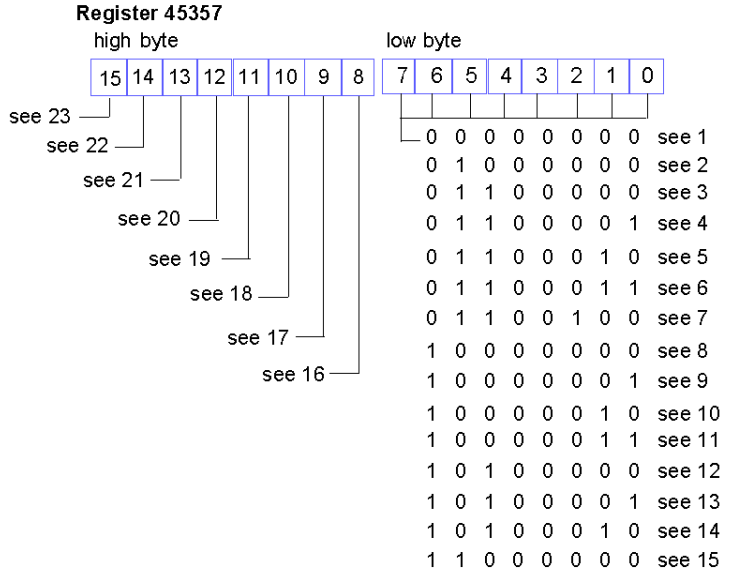
Predefined Diagnostics Registers in the Data Image

Summary

Thirty-five contiguous registers (45357 through 45391) in the island bus data image *(see page 142)* are provided for reporting diagnostic information. These registers have predefined meanings that are described below. The numerical values associated with each message can be accessed and monitored with an HMI panel. The messages themselves appear in the log window and in other displays in the Advantys configuration software.

Island Communications Status

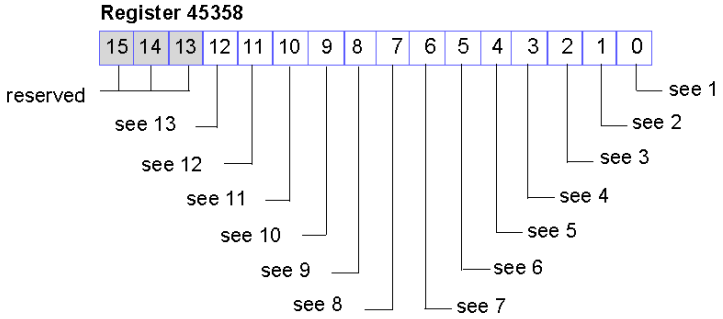
Register 45357 describes the state of communications across the island bus. The low byte (bits 7 through 0) displays one of 15 possible eight-bit patterns that indicates the current state of communication. Each bit in the high byte (bits 15 through 8) is used to signal the presence or absence of a specific error condition:



1 The island is initializing.
2 The island has been put in the pre-operational state, for example, with the reset function in the Advantys STB configuration software.
3 The NIM is configuring or auto-configuring—communication to all modules is reset.
4 The NIM is configuring or auto-configuring—checking for any modules that are not auto-addressed.
5 The NIM is configuring or auto-configuring—Advantys STB and preferred modules are being auto-addressed.
6 The NIM is configuring or auto-configuring—boot-up is in progress.
7 The process image is being set up.
8 Initialization is complete, the island bus is configured, the configuration matches, and the island bus is not started.
9 Configuration mismatch—non-mandatory or unexpected modules in the configuration do not match, and the island bus is not started.
10 Configuration mismatch—at least one mandatory module does not match, and the island bus is not started.
11 Serious configuration mismatch—the island bus has been set to pre-operational mode, and initialization is aborted.
12 The configuration matches, and the island bus is operational.
13 Island is operational with a configuration mismatch. At least one standard module does not match, but all the mandatory modules are present and operating.
14 Serious configuration mismatch—the island bus was started but is now in pre-operational mode because of one or more mismatched mandatory module(s).
15 Island has been set to pre-operational mode, for example, with the stop function in the Advantys STB configuration software.
16 A value of 1 in bit 8 is a fatal error. It indicates a low-priority receive queue software overrun error.
17 A value of 1 in bit 9 is a fatal error. It indicates a NIM overrun error.
18 A value of 1 in bit 10 indicates an island bus-off error.
19 A value of 1 in bit 11 is a fatal error. It indicates that the error counter in the NIM has reached the warning level and the error status bit has been set.
20 A value of 1 in bit 12 indicates that the NIM's error status bit has been reset.
21 A value of 1 in bit 13 is a fatal error. It indicates a low-priority transfer queue software overrun error.
22 A value of 1 in bit 14 is a fatal error. It indicates a high-priority receive queue software overrun error.
23 A value of 1 in bit 15 is a fatal error. It indicates a high-priority transfer queue software overrun error.

Error Reporting

Each bit in register 45358 is used to report a global error condition. A value of 1 in the bit indicates that a specific global error has been detected:



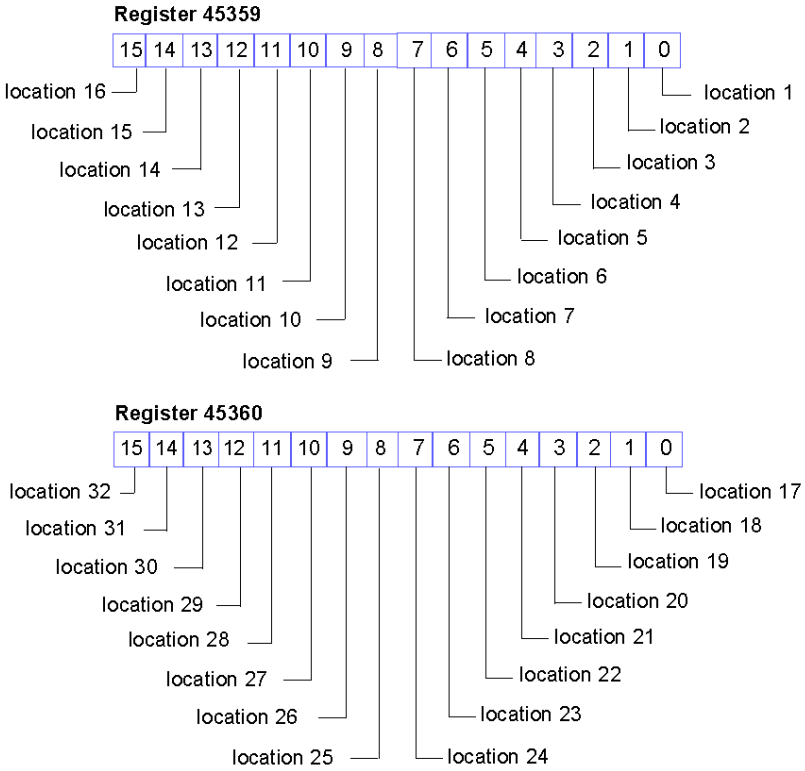
1 Fatal error. Because of the severity of the error, no further communications are possible on the island bus.
2 Module ID error—a standard CANopen device is using a module ID reserved for the Advantys STB modules.
3 Auto-addressing has failed.
4 Mandatory module configuration error.
5 Process image error—either the process image configuration is inconsistent, or it could not be set up during auto-configuration.
6 Auto-configuration error—a module is not in its configured location, and the NIM cannot complete auto-configuration.
7 Island bus management error detected by the NIM.
8 Assignment error—the initialization process in the NIM has detected a module assignment error, possibly the result of at least one application parameter mismatch.
9 Internal triggering protocol error.
10 Module data length error.
11 Module configuration error.
12 Application parameter error.
13 Application parameter services or timeout error.

Node Configuration

The next eight contiguous registers (registers 45359 through 45366) display locations where modules have been configured on the island bus. This information is stored in Flash memory. At start up, the actual locations of the modules on the island are validated by comparing them to the configured locations stored in memory. Each bit represents a configured location:

- A value of 1 in a bit indicates that a module has been configured for the associated location.
- A value of 0 in a bit indicates that a module has not been configured for the associated location.

The first two registers, shown below, provide the 32 bits that represent the module locations in a typical island configuration. The remaining six registers (45361 through 45366) are available to support island expansion capabilities.

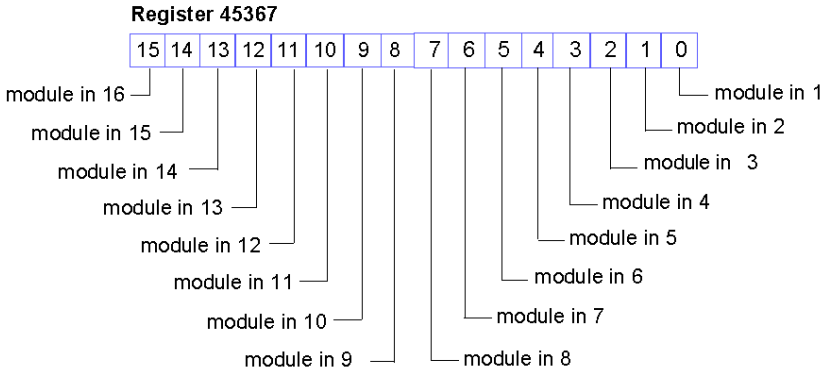


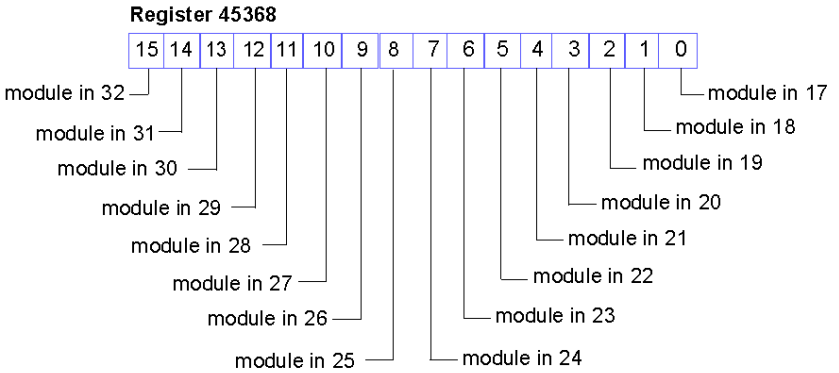
Node Assembly

The next eight contiguous registers (registers 45367 through 45374) indicate the presence or absence of configured modules in locations on the island bus. This information is stored in Flash memory. At start up, the actual locations of the modules on the island are validated by comparing them to the configured locations stored in memory. Each bit represents a module:

- A value of 1 in a given bit indicates either that the configured module is not present or that the location has not been configured.
- A value of 0 indicates that the correct module is present in its configured location.

The first two registers, shown below, provide the 32 bits that represent the module locations in a typical island configuration. The remaining six registers (45369 through 45374) are available to support island expansion capabilities.



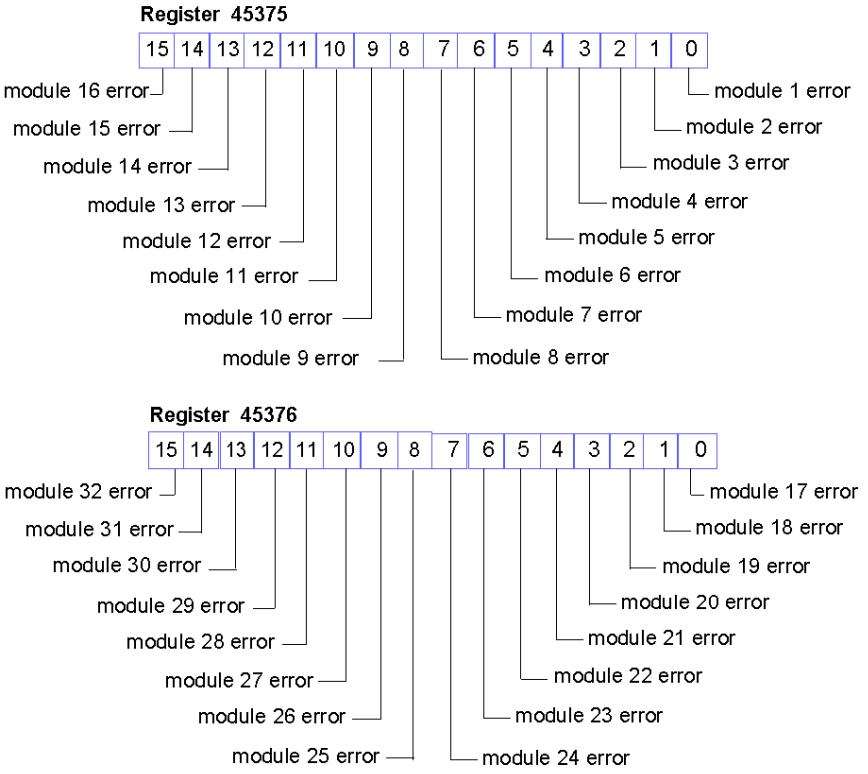


Emergency Messages

The next eight contiguous registers (registers 45375 through 45382) indicate the presence or absence of newly received emergency messages for individual modules on the island. Each bit represents a module:

- A value of 1 in a given bit indicates that a new emergency message has been queued for the associated module.
- A value of 0 in a given bit indicates that no new emergency messages have been received for the associated module since the last time the diagnostic buffer was read.

The first two registers, shown below, provide the 32 bits that represent the module locations in a typical island configuration. The remaining six registers (45377 through 45382) are available to support island expansion capabilities.

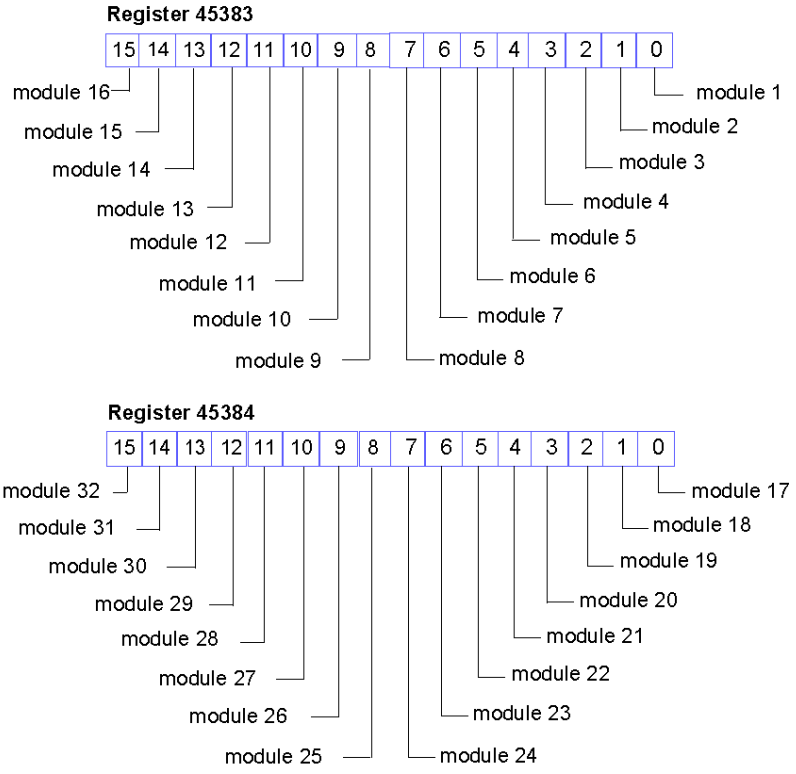


Fault Detection

The next eight contiguous registers (registers 45383 through 45390) indicate the presence or absence of operational faults detected on the island bus modules. Each bit represents a module:

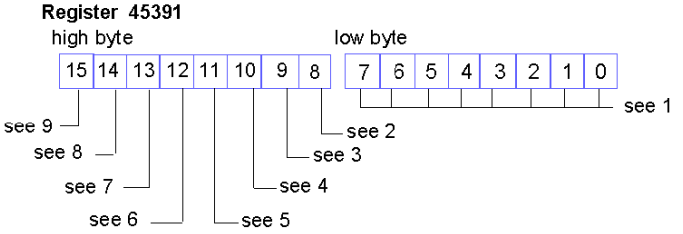
- A value of 1 in a bit indicates that the associated module is operating and that no faults were detected.
- A value of 0 in a bit indicates that the associated module is not operating either because it has a fault or because it has not been configured.

The first two registers, shown below, provide the 32 bits that represent the module locations in a typical island configuration. The remaining six registers (45385 through 45390) are available to support island expansion capabilities.



NIM Status

The eight LSBs (bits 8 through 15) in register 45391 report the status of the CANopen NIM. The eight MSBs (bits 7 though 0) are always zeros:



1 Fieldbus-dependent.
2 Module failure—bit 8 is set to 1 if any module on the island bus fails.
3 A value of 1 in bit 9 indicates an internal failure—at least one global bit is set.
4 A bit value of 1 in bit 10 indicates an external failure—the problem is on the fieldbus.
5 A value of 1 in bit 11 indicates that the configuration is protected—the RST button is disabled and the configuration software requires a password before you can write. A bit value of 0 indicates that the configuration is standard—the RST button is enabled and the configuration software is not password-protected.
6 A value of 1 in bit 12 indicates that the configuration on the replaceable memory card is invalid.
7 A value of 1 in bit 13 indicates that reflex action funcitonality has been configured. (For NIMs with firmware version 2.0 or higher.)
8 A value of 1 in bit 14 indicates that one or more island modules have been hot-swapped. (For NIMs with firmware version 2.0 or higher.)
9 Island bus output data master—A value of 0 in bit 15 indicates that the fieldbus master device is controlling the output data of the island's process image; a bit value of 1 indicates that the Advantys configuration software is controlling the output data of the island's process image.

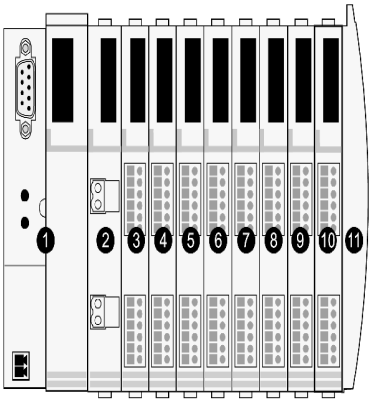
An Example of a Modbus View of the Process Image

Summary

The following example shows what the output data process image and the input data and I/O status process image might look like when it represents a specific island bus configuration.

The Sample Configuration

The sample island comprises the following 10 modules and a termination plate:



1 network interface module
2 24 VDC power distribution module
3 STB DDI 3230 24 VDC two-channel digital input module
4 STB DDO 3200 24 VDC two-channel digital output module
5 STB DDI 3420 24 VDC four-channel digital input module
6 STB DDO 3410 24 VDC four-channel digital output module
7 STB DDI 3610 24 VDC six-channel digital input module
8 STB DDO 3600 24 VDC six-channel digital output module
9 STB AVI 1270 +/-10 VDC two-channel analog input module
10 STB AVO 1250 +/-10 VDC two-channel analog output module
11 STB XMP 1100 island bus termination plate

The I/O modules have the following island bus addresses *(see page 48)*:

I/O Model	Module Type	Island Bus Address
STB DDI 3230	two-channel digital input	1
STB DDO 3200	two-channel digital output	2
STB DDI 3420	four-channel digital input	3
STB DDO 3410	four-channel digital output	4
STB DDI 3610	six-channel digital input	5

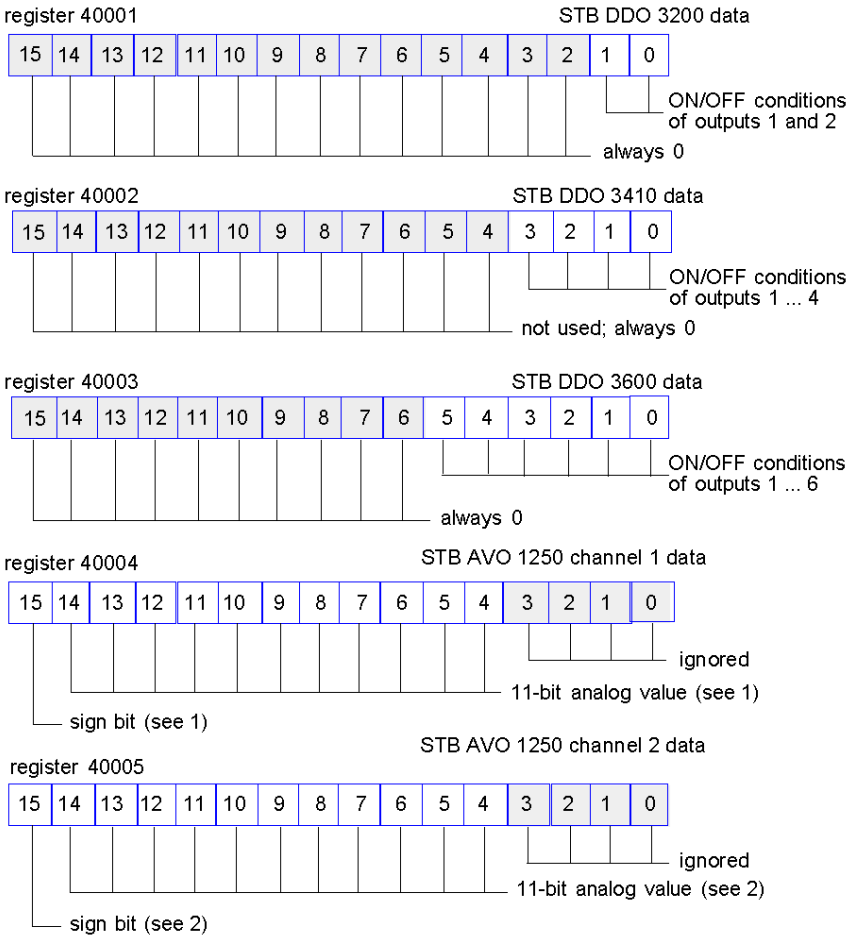
I/O Model	Module Type	Island Bus Address
STB DDO 3600	six-channel digital output	6
STB AVI 1270	two-channel analog input	7
STB AVO 1250	two-channel analog output	8

The PDM and the termination plate do not consume island bus addresses and are not represented in the process image.

The Output Data Process Image

Let's look first at the register allocation required to support the output data process image *(see page 144)*. This is the data written to the island from the fieldbus master to update the output modules on the island bus. The four output modules are affected—the three digital output modules at addresses 2, 4, and 6 and the one analog output module at address 8.

The three digital output modules utilize one Modbus register apiece for data. The analog output module requires two registers, one for each output channel. A total of five registers (registers 40001 through 40005) are used for this configuration:



1 The value represented in register 40004 is in the range +10 to -10 V, with 11-bit resolution plus a sign bit in bit 15.

2 The value represented in register 40005 is in the range +10 to -10 V, with 11-bit resolution plus a sign bit in bit 15.

The digital modules use the LSBs to hold and display their output data. The analog module uses the MSBs to hold and display its output data.

The Input Data and I/O Status Process Image

Now let's look at the register allocation required to support the input data and I/O status process image *(see page 145)*. This is the information that the NIM collects from the island modules so that it can be read by the fieldbus master or by some other monitoring device.

All eight I/O modules are represented in this process image block. The modules are assigned registers in the order of their island bus addresses, starting at register 45392.

Each digital I/O module uses two contiguous registers:

- Digital input modules use one register to report data and the next to report status.
- Digital output modules use one register to echo output data and the next to report status.

NOTE: The value in an *echo output data* register is basically a copy of the value written to the corresponding register in the output data process image. Generally, this is the value written to the NIM by the fieldbus master, and its echo is of not much interest. When an output channel is configured to perform a reflex action *(see page 133)*, however, the echo register provides a location where the fieldbus master can look to see the current value of the output.

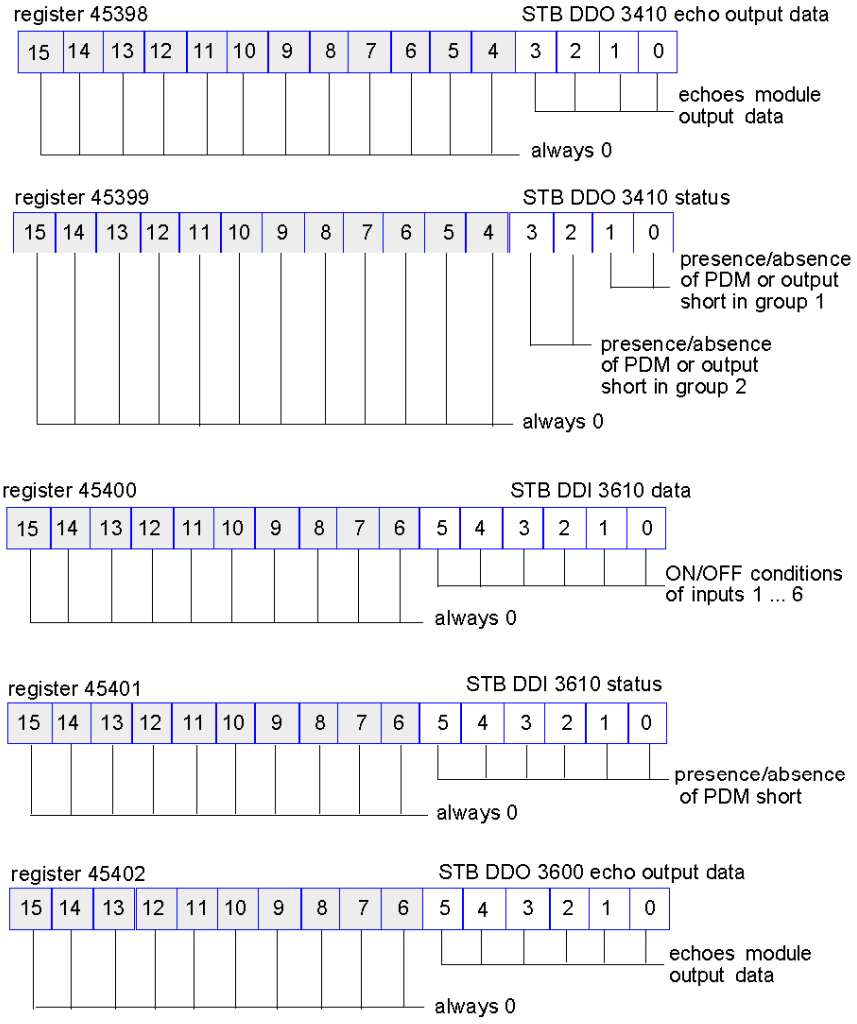
The analog input module uses four contiguous registers:

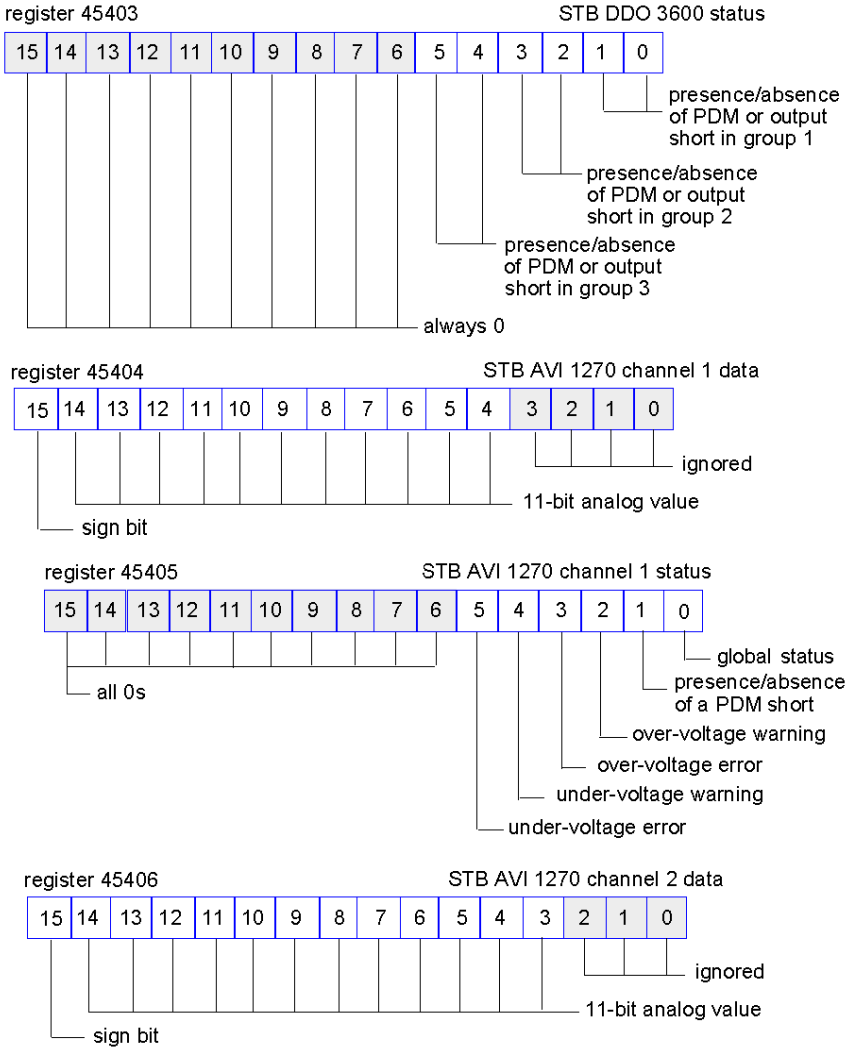
- the first register to report the data for channel 1
- the second register to report status for channel 1
- the third register to report the data for channel 2
- the fourth register to report status for channel 2

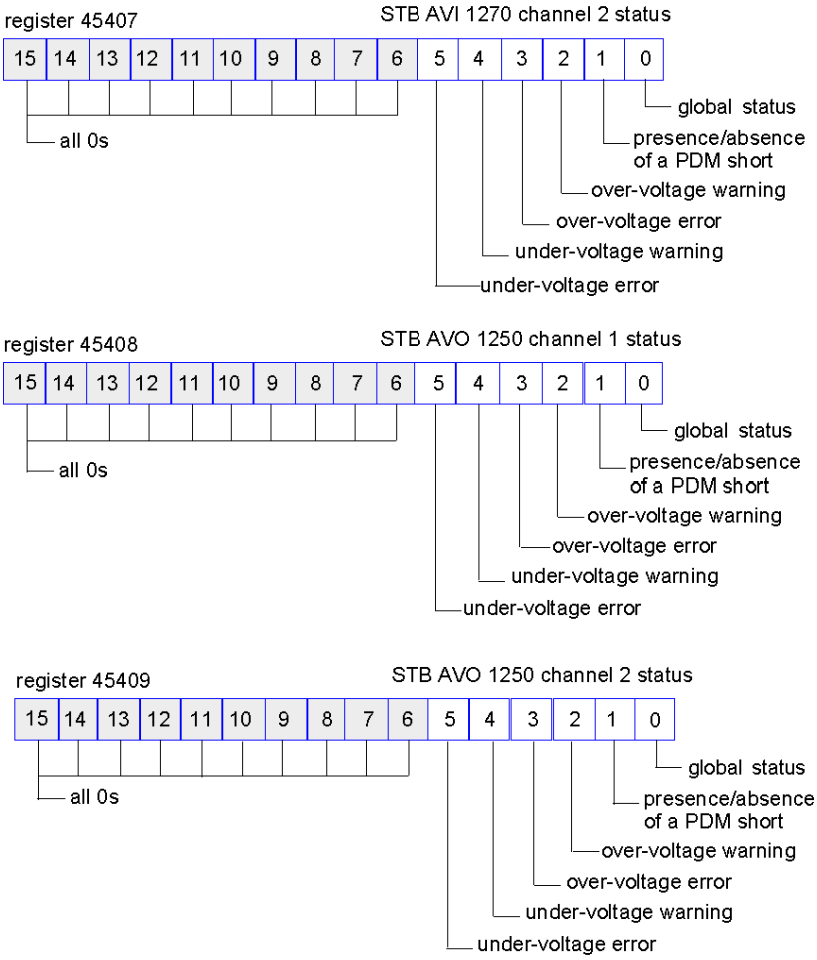
The analog output module uses two contiguous registers:

- the first register to report status for channel 1
- the second register to report status for channel 2

In total, 18 registers (registers 45392 through 45409) are used to support our configuration:


register 45398
STB DDO 3410 echo output data
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
echoes module output data
always 0
register 45399
STB DDO 3410 status
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
presence/absence of PDM or output short in group 1
presence/absence of PDM or output short in group 2
always 0
register 45400
STB DDI 3610 data
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ON/OFF conditions of inputs 1 ... 6
always 0
register 45401
STB DDI 3610 status
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
presence/absence of PDM short
always 0
register 45402
STB DDO 3600 echo output data
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
echoes module output data
always 0


register 45403
STB DDO 3600 status
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
presence/absence of PDM or output short in group 1
presence/absence of PDM or output short in group 2
presence/absence of PDM or output short in group 3
always 0
register 45404
STB AVI 1270 channel 1 data
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ignored
11-bit analog value
sign bit
register 45405
STB AVI 1270 channel 1 status
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
global status
presence/absence of a PDM short
over-voltage warning
over-voltage error
under-voltage warning
under-voltage error
all 0s
register 45406
STB AVI 1270 channel 2 data
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ignored
11-bit analog value
sign bit


register 45407
STB AVI 1270 channel 2 status
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
all 0s
global status
presence/absence of a PDM short
over-voltage warning
over-voltage error
under-voltage warning
under-voltage error
register 45408
STB AVO 1250 channel 1 status
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
all 0s
global status
presence/absence of a PDM short
over-voltage warning
over-voltage error
under-voltage warning
under-voltage error
register 45409
STB AVO 1250 channel 2 status
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
all 0s
global status
presence/absence of a PDM short
over-voltage warning
over-voltage error
under-voltage warning
under-voltage error

The HMI Blocks in the Island Data Image

Summary

An HMI panel that communicates using the Modbus protocol can be connected to the CFG port *(see page 35)* on the NIM. Using the Advantys configuration software, you can reserve one or two blocks of registers in the data image *(see page 141)* to support HMI data exchange. When an HMI panel writes to one of these blocks, that data is accessible to the fieldbus master (as inputs). Data written by the fieldbus master (as outputs) is stored in a different reserved block of registers that the HMI panel can read.

HMI Panel Configuration

Advantys STB supports the ability of an HMI panel to act as:

- an input device, which writes data to the island's data image that is read by the fieldbus master
- an output device, which can read data written by the fieldbus master to the island's data image
- a combined I/O device

HMI Input Data Exchange

Input data to the fieldbus master can be generated by the HMI panel. Input controls on an HMI panel might be elements such as:

- push buttons
- switches
- a data entry keypad

To use an HMI panel as an input device on the island, you need to enable the HMI-to-fieldbus master block in the island's data image *(see page 142)* and specify the number of registers in this block that you want to use for HMI-to-fieldbus master data transfers. You must use the Advantys configuration software to make these configuration adjustments.

The HMI-to-fieldbus master block can comprise up to 512 registers, ranging from register 49488 to 49999. (Your actual register limit will be dictated by your fieldbus.) This block follows immediately after the standard input data and I/O status process image *(see page 145)* block (registers 45392 through 49487) in the island's data image.

The HMI panel writes the input data to a specified number of registers in the HMI-to-fieldbus master block. The NIM manages the transfer of the HMI data in these registers as part of the overall input data transfer—it converts the 16-bit register data to a fieldbus-specific data format and transfers it together with the standard input data and I/O status process image to the fieldbus. The fieldbus master sees and responds to HMI data as if it were standard input data.

HMI Output Data Exchange

In turn, output data written by the fieldbus master can be used to update enunciator elements on the HMI panel. Enunciator elements might be:

- display indicators
- buttons or screen images that change color or shape
- data display screens (for example, temperature read-outs)

To use the HMI panel as an output device, you need to enable the fieldbus-to-HMI block in the island's data image *(see page 142)* and specify the number of registers in this block that you want to use. You need to use the Advantys configuration software to make these adjustments to your configuration.

The fieldbus master-to-HMI block can comprise up to 512 registers, ranging from register 44097 to 44608. This block follows immediately after the standard output data process image *(see page 144)* block (registers 40001 through 44096) in the island's data image.

The fieldbus master writes output update data in native fieldbus format to the HMI data block concurrent with writing this data to the output data process image area. The output data is placed in the fieldbus master-to-HMI block. Upon request by the HMI via a Modbus *read* command, the role of the NIM is to receive this output data, convert it to16-bit Modbus format, and send it over the Modbus connection at the CFG port to the HMI panel.

NOTE: The *read* command enables all Modbus registers to be read, not just those in the block reserved for fieldbus master-to-HMI data exchange.

Test Mode

Summary

Test Mode indicates that the output data of the STB island's process image is not controlled by a fieldbus master device, but is instead controlled by either the Advantys Configuration Software or an HMI. When the STB island is operating in Test Mode, the fieldbus master cannot write the STB island's outputs, but can continue to read its inputs and diagnostic data.

Test Mode is configured off-line, downloaded with the island configuration, then activated online.

Select Test Mode Settings in the **Online** menu to open the Test Mode configuration window, where you can select a test mode setting. Test Mode settings are stored with other STB island configuration settings both in the NIM's flash memory and in a SIM card, if one is attached to the NIM.

When Test Mode is activated, the NIM's TEST LED is lit, and bit #5 of the NIM Status word in register 45391 is set to 1.

NOTE: Loss of Modbus communications do not affect Test Mode.

There are three Test Mode settings:

- Temporary Test Mode
- Persistent Test Mode
- Password Test Mode

The following sections describe the process and effect of activating Test Mode.

Temporary Test Mode

When operating online, use the Advantys Configuration Software (not an HMI) to activate Temporary Test Mode, by selecting **Test Mode** in the **Online** menu.

Once activated, Temporary Test Mode is deactivated by:

- de-selecting **Test Mode** in the **Online** menu
- cycling power to the NIM
- selecting **Reset** in the **Online** menu
- performing Autoconfiguration
- downloading a new island configuration to the NIM (or inserting a SIM card with a new island configuration into the NIM and cycling power to the NIM).

Temporary Test Mode is the default Test Mode configuration setting.

Persistent Test Mode

Use the Advantys Configuration Software to configure the STB island for Persistent Test Mode. When the download of this configuration is complete, Persistent Test Mode is activated. Thereafter, the STB island operates in Test Mode each time power is cycled to the island. When Persistent Test Mode is activated, the STB island's process image output data is controlled exclusively by either the HMI or the configuration software. The fieldbus master no longer controls these outputs.

Persistent Test Mode is deactivated by:

- downloading a new island configuration to the NIM (or inserting a SIM card with a new island configuration into the NIM and cycling power to the NIM)
- performing Autoconfiguration.

Password Test Mode

Use the Advantys Configuration Software to enter a password to the STB island's configuration settings. The password you input must have an integer value from 1 to 65535 (FFFF hex).

After the changed configuration (including the password) has been downloaded, you can activate Password Test Mode only by using an HMI to issue a single Modbus Register write command to send the password value to Modbus Register 45120.

After Password Test Mode is activated, the STB island's process image output data is controlled by either the HMI or the configuration software. In this case, the fieldbus master no longer controls these outputs.

Password Test Mode, once activated, is deactivated by:

- cycling power to the NIM
- selecting **Reset** in the **Online** menu
- performing Autoconfiguration
- downloading a new island configuration to the NIM (or inserting a SIM card with a new island configuration into the NIM and cycling power to the NIM)
- using an HMI to issue a single Modbus register write command to send the password value to Modbus Register 45121 (STB NIC 2212 and STB NIP 2311 NIMs only)

NOTE: Password Test Mode must be activated only by using the NIM's configuration port. All attempts to enter Password Test Mode using the fieldbus (via NIM models STB NMP 2212 or STB NIP 2212) are unsuccessful.

Run-Time Parameters

Introduction

For STB modules, the Advantys Configuration Software provides the RTP (run-time parameters) feature. It can be used for monitoring and modifying selected I/O parameters and Island bus status registers of the NIM while the Island is running. This feature is available only in standard STB NIMs with firmware version 2.0 or later.

RTP must be configured using the Advantys Configuration Software before it can be used. RTP is not configured by default. Configure RTP by selecting **Configure run-time Parameters** in the **Options** tab of the NIM Module Editor. This allocates the necessary registers within the NIM's data process image to support this feature.

Request and Response Blocks

Once configured, use the RTP feature by writing up to 5 reserved words in the NIM's output data process image (the RTP request block) and by reading the value of 4 reserved words in the NIM's input data process image (the RTP response block). The Advantys Configuration Software displays both blocks of reserved RTP words in the Island's **I/O Image Overview** dialog box, both in the **Modbus Image** tab and (for NIMs with a separate fieldbus image) in the **Fieldbus Image** tab. In each tab, the blocks of reserved RTP words appear after the block of process I/O data and before the block of HMI data (if any).

NOTE: The Modbus address values of the RTP request and response blocks are the same in all standard NIMs. The fieldbus address values of the RTP request and response blocks depend upon the network type. Use the **Fieldbus Image** tab of the **I/O Image Overview** dialog box to obtain the location of the RTP registers. For Modbus Plus and Ethernet networks, use the Modbus register numbers.

Exceptions

Any parameter you modify using the RTP feature does not retain its modified value if one of the following events occurs:

- Power is cycled to the NIM.
- A **Reset** command is issued to the NIM using the Advantys Configuration Software.
- A **Store to SIM Card** command is issued using the Advantys Configuration Software.
- The module whose parameter has been modified is hot-swapped.
  If a module is hot-swapped, as indicated by the HOT_SWAP indicator bit, you can use the RTP feature to detect which module has been hot-swapped and to restore the parameters to their previous values.

Test Mode

When the NIM is operating in test mode, the NIM's output data process image (including the RTP request block) can be controlled either by the Advantys Configuration Software or by an HMI (depending upon the test mode configured). Standard Modbus commands can be used to access the RTP words. If the NIM is in test mode, the fieldbus master cannot write to the RTP request block in the NIM's output data process image.

RTP Request Block Words Definitions

The following table lists RTP request block words:

Modbus Address	Upper Byte	Lower Byte	Data Type	Attribute
45130	sub-index	`toggle + length`	unsigned 16	RW
45131	index (high data byte)	index (low data byte)	unsigned 16	RW
45132	data byte 2	data byte 1 (LSB)	unsigned 16	RW
45133	data byte 4 (MSB)	data byte 3	unsigned 16	RW
45134	`toggle + CMD`	Node ID	unsigned 16	RW
NOTE: The RTP request block is also presented in the manufacturer specific area of the CANopen fieldbus as an object with a dedicated index of 0x4101 and sub-index 1 to 5 (data type = unsigned 16, attribute = RW).				

The NIM performs range checking on the above bytes as follows:

- index (high / low byte): 0x2000 to 0xFFFF for write; 0x1000 to 0xFFFF for read
- `toggle + length`: length = 1 to 4 bytes; the most significant bit contains the toggle bit
- `toggle + CMD`: CMD = 1 to 0x0A (see the table *Valid Commands*, below); most significant bit contains toggle bit
- Node ID: 1 to 32 and 127 (the NIM itself)

The `toggle+CMD` and `toggle+length` bytes are at either end of the RTP request register block. The NIM processes the RTP request when the same value is set in the respective toggle bits of these two bytes. The NIM processes the same RTP block again only when both values have changed to a new identical value. We recommend that you configure new matching values for the two toggle bytes (`toggle+CMD` and `toggle+length`) only after you have constructed the RTP request between them.

⚠ WARNING
UNINTENDED EQUIPMENT OPERATION Write all bytes in the RTP request before you set the `toggle+CMD` and `toggle+length` bytes to the same new value. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

RTP Response Block Words Definitions

The following list shows RTP response block words:

Modbus Address	Upper Byte	Lower Byte	Data Type	Attribute
45303	status (the most significant bit is used to indicate whether RTP service is enabled: MSB=1 means enabled)	`toggle + CMD echo`	unsigned 16	RO
45304	data byte 2	data byte 1 (LSB)	unsigned 16	RO
45305	data byte 4 (MSB)	data byte 3	unsigned 16	RO
45306	-	`toggle + CMD echo`	unsigned 16	RO
NOTE: The RTP response block is also presented in the manufacturer specific area of the CANopen fieldbus as an object with a dedicated index of 0x4100 and sub-index 1 to 4 (data type = unsigned 16, attribute = RO).				

The `toggle+CMD echo` bytes are located at the end of the register range to let you validate the consistency of the data wrapped within these bytes (in case RTP response block words are not updated in a single scan). The NIM updates the status byte and the 4 data bytes (if applicable) before updating the `toggle+CMD echo` bytes in Modbus register 45303 and 45306 to equal the value of the `toggle+CMD` byte of the corresponding RTP request. You must first check that both `toggle+CMD` bytes match the `toggle+CMD` byte in the RTP request block before making use of the data inside the RTP response block.

Valid RTP Commands

The following list shows valid commands (CMDs):

Command (CMD)	Code (Except the msb)	Valid Node IDs	Allowed State of the Addressed Node	Data Bytes
Enable RTP (Only After RTP Has Been Configured Using the Advantys Configuration Software)	0x08	127	N/A	-
Disable RTP	0x09	127	N/A	-
Reset Hot-Swap Bit	0x0A	1-32	N/A	-
Read Parameter	0x01	1-32, 127	pre-operational operational	data bytes in response, length to be given
Write Parameter	0x02	1-32	operational	data bytes in request, length to be given

The most significant bit of an RTP request block's `toggle+CMD` byte is the toggle bit. A new command is identified when the value of this bit changes and matches the value of the toggle bit in the `toggle+length` byte.

A new RTP request is processed only if the preceding RTP request has finished. Overlapping RTP requests are not allowed. A new RTP request made before the completion of a preceding request is ignored.

To determine when an RTP command has been processed and its response is complete, check the values of the `toggle+CMD echo` bytes in the RTP response block. Continue to check both `toggle+CMD` bytes in the RTP response block until they match the RTP request block's `toggle+CMD` byte. Once they match, the contents of the RTP response block is valid.

Valid RTP Status Messages

The following list shows valid status messages:

Status Byte	Code	Comment
Success	0x00 or 0x80	0x00 for successful completion of a Disable RTP command
Command not Processed due to Disabled RTP	0x01	-
Illegal CMD	0x82	-
Illegal Data Length	0x83	-
Illegal Node ID	0x84	-
Illegal Node State	0x85	Access is denied because a node is absent or not started.
Illegal Index	0x86	-
RTP Response Has More Than 4 Bytes	0x87	-
No Communication Possible on the Island Bus	0x88	-
Illegal Write to Node 127	0x89	-
SDO Aborted	0x90	If an SDO protocol error is detected, the data bytes in the response contain the SDO abort code according to DS301.
General Exception Response	0xFF	This is a status event of a type other than those specified above.

The most significant bit of the status byte in the RTP response block indicates whether RTP is enabled (1) or disabled (0).

Virtual Placeholder

Summary

The virtual placeholder feature lets you create a standard island configuration and depopulated variations of that configuration that share the same fieldbus process image, thereby letting you maintain a consistent PLC or fieldbus master program for various island configurations. The depopulated islands are physically built using only those modules that are not marked as *not present*, thus saving cost and space.

As part of an Advantys STB island custom configuration, you can set *Virtual Placeholder* status for any STB I/O or preferred third-party module whose node address is assigned by the NIM during auto-addressing.

After a module has been assigned Virtual Placeholder status, you can physically remove it from its Advantys STB island base, while retaining the island's process image. All modules that physically remain in the Advantys STB island configuration will retain their previous node addresses. This lets you physically alter the design of your island, without having to edit your PLC program.

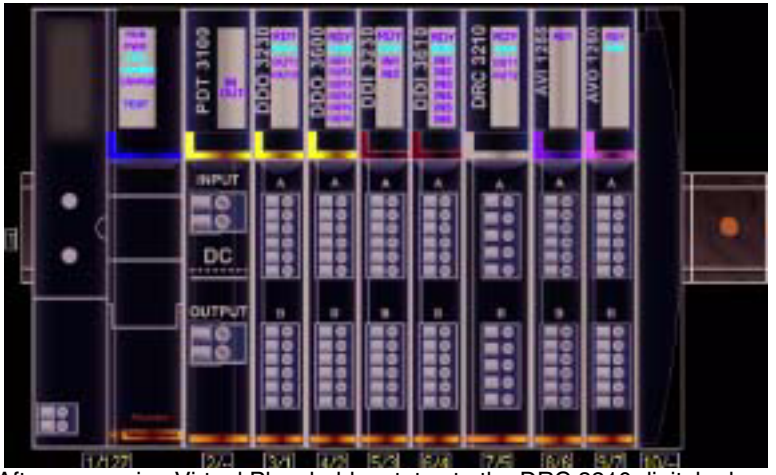
NOTE: Advantys configuration software is required to set Virtual Placeholder status.

Setting Virtual Placeholder Status

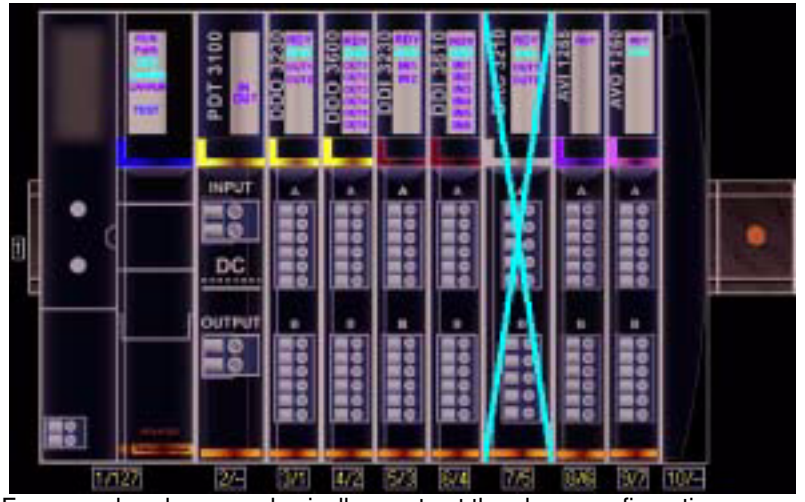
To set Virtual Placeholder status:

Step	Action
1	Open the STB I/O or preferred third-party module's property window.
2	In the Options tab, select **Not Present**.
3	Click **OK** to save your settings. The Advantys STB configuration software marks the virtual placeholder module with a red "X" (as shown below).

For example, the following island configuration contains a NIM, a PDM, 2 digital Input modules, 2 digital output modules, a digital relay output module, an analog input module, and an analog output module:



After you assign Virtual Placeholder status to the DRC 3210 digital relay output module (by selecting **Not Present** in its Options tab), the Advantys STB configuration software marks the virtual placeholder module with a red "X" as shown below:



For example, when you physically construct the above configuration, you would build the island without the DRC-3210 and its base.

NOTE: Any reflex output, that is configured to use a virtual placeholder module as an input, will constantly be in fallback.

The Remote Virtual Placeholder Option: Overview

Summary

A limitation of the standard virtual placeholder capability *(see page 171)* is the need to configure and maintain a separate configuration (or view of the process image) for every variation in the island's physical layout. You need to download a different view with the Advantys configuration software each time you want to change a virtual placeholder configuration on the island bus.

With the *remote* virtual placeholder option, you create one fully defined process image that contains all the I/O modules you need for all the desired views of the physical island. The fieldbus master then manages the reconfiguration change remotely. The fieldbus does this by writing a valid reconfiguration to a special remote virtual placeholder object in the island's CANopen object dictionary *(see page 71)*.

Valid Configurations

A valid remote virtual placeholder configuration can comprise any combination of up to 32 I/O modules on the island bus as long as:

- any modules that are declared not present are Advantys STB I/O modules or preferred modules
- the remote virtual placeholder configuration accurately reflects the true module population on the physical island

Devices on a CANopen extension of the island bus cannot be defined as not present in a remote virtual placeholder configuration. If you attempt to define a CANopen extension module as not present, the transaction will report an error in the IOS object *(see page 177)* and the reconfiguration attempt will fail.

Software Considerations

The remote virtual placeholder option is available in version 2.2 or greater of the Advantys configuration software.

Once you select the remote virtual placeholder option with the Advantys configuration software, the software can monitor and control the island bus but it does not participate in the writing of remote virtual placeholder information to the island. The standard virtual placeholder capability is disabled, and you cannot configure any modules as not present with the Advantys configuration software.

NIM Firmware Requirements

The firmware in the STBNCO2212 NIM must be at version 3.*x* or greater to support the remote virtual placeholder option. Version 3.*x* firmware is compatible with earlier versions of the NIM. Firmware updates can be installed with the firmware loader utility, which is distributed with the Advantys configuration software.

Existing projects created with earlier versions of the Advantys configuration software (before version 3.*x*) can be downloaded to later versions of the NIM. The projects may be downloaded either unchanged or modified and rebuilt.

Use of the Removable Memory Card

An STB XMP 4440 removable memory card *(see page 55)* can store a configuration that has the remote virtual placeholder capability enabled.

NOTE: The full configuration is always stored to the memory card with the virtual placeholder capability enabled in the NIM. You cannot store a configuration with placeholder modules configured as not present on the memory card.

If a memory card with a remote virtual placeholder configuration on it is inserted in a version 2.*x* NIM, the configuration on the card is accepted but the remote virtual placeholder capability is disabled.

Reconfiguring the Island at Initial Start-up

To reconfigure the island with the remote virtual placeholder capability, the fieldbus must write a new configuration data to a subindex in the VPCW object *(see page 178)*, then issue two requests—a reconfiguration request followed by a start request. The following table describes the sequence of interactions between the fieldbus master and the NIM at initial start-up. A more detailed application example is also provided in the appendix *(see page 183)*.

Stage	The Fieldbus Master ...	The NIM ...
1	... waits for a connection to the NIM. The program controlling the fieldbus master needs to monitor the IOS object, waiting for a value of 0001 hex (indicating that the NIM has a configuration and is ready to run).	... requests that the island bus boot up, initializes the island's object dictionary, establishes communication with the fieldbus, then sets the value in the IOS object to 0001 hex.
2	... writes a new remote virtual placeholder configuration to VPCW object subindex 1.	
3	... sends a reconfiguration request to the NIM by setting the IOC object to a value of 0001 hex.	... sets the status island to busy (by setting the value of the IOS object to 0000 hex) and initializes island bus communications. Then the NIM stores the configuration values from VPCW subindex 1 in Flash. After this, it requests that the island bus boot up again and sets the value in the IOS object to 0001 hex. **Note** The store-to-Flash operation may take several seconds to complete (typically from 7 to 10 s). Once the reconfiguration request is complete and accepted, the NIM will use the configuration in Flash unless and until it receives a new reconfiguration request.

Stage	The Fieldbus Master ...	The NIM ...
4	... monitors the IOS object for a value of 0001 hex, then sends a start request to the island (by writing a value of 0002 hex to the IOC object).	... sets the value of the IOS object to 0000 hex (busy), puts the island bus into run mode, then sets the IOS object to 0002 hex (indicating that the start request was processed successfully).
5	The island starts running and exchanging data with the fieldbus master.	

Restarting an Island after a Reconfiguration

The island cannot start or restart automatically when the remote virtual placeholder option is enabled in the configuration downloaded by the Advantys configuration software. The island must be restarted by the fieldbus master or, in some cases, by the Advantys configuration software. The following table describes the sequence of interactions between the fieldbus master and the NIM whenever the island is restarted. The fieldbus master sends an explicit start request to put the island into run mode:

Stage	The Fieldbus Master ...	The STBNCO2212 NIM ...
1	... waits for a connection to the STBNCO2212 NIM. It monitors the IOS object, waiting for a value of 0001 hex (indicating that the NIM has a configuration and is ready to run).	... requests that the island bus boot up, initializes the island's object dictionary, establishes communication with the fieldbus, then sets the value in the IOS object to 0001 hex.
2	... checks subindex 1 of the NIM's VPCR object to determine that the current configuration is correct.	
3	... sends a start request to the island by writing a value of 0002 hex to the IOC object.	... sets the status island to busy (by setting the value of the IOS object to 0000 hex), puts the island bus into run mode, then sets the IOS object to 0002 hex (indicating that the start request was processed successfully).
4	... checks for a value of 0002 hex in the IOC object).	
5	The island starts running and exchanging data with the fieldbus master.	

Handling Multiple Requests

The remote virtual placeholder capability is designed to handle one request at a time. It is recommended that you design your program so that the fieldbus master does not stack multiple requests before the completion of one request. For example, after issuing a reconfiguration request the fieldbus master should check the status in the IOS object to make sure the request has been processed before it issues a start request. If the start request is issued while the reconfiguration request is still being processed, the start request may be lost.

Simultaneous Access to the Island Bus

Both the fieldbus master and the Advantys configuration software (in online mode) can control the island. Both entities can access the island simultaneously, and either entity may start the island except in the case of test mode.

If you use the Advantys configuration software to take the island out of run mode, then you must use the software to restart the island. Commands from the fieldbus are not executed.

Because the Advantys configuration software can take control of an island that is operating with a remote virtual placeholder configuration, it can invoke changes in the island that may not be reflected in the IOS status object *(see page 177)*. For example, the IOS object may report that it has received a start request from the fieldbus master and has started the island with a newly written configuration. If the Advantys configuration software later takes the island offline and puts it in test mode, the IOS object still reports that the start request has been successfully processed.

Special Objects for the Remote Virtual Placeholder Option

Summary

When the remote virtual placeholder is enabled, 4 special objects appear in the CANopen fieldbus object dictionary that support this remote configuration capability.

- the island operation control (IOC) object, which is the mechanism by which the fieldbus master sends control requests to the NIM
- the island operation status (IOS) object, which reports the status of these control requests when they are executing successfully and reports errors when the requests are rejected
- the virtual placeholder configuration write (VPCW) object, which provides two 32-bit subindices where fieldbus can write the desired reconfiguration information; a module that is expected to be present at a location on the physical island is represented by a 0 and a logical node that is expected to be not present on the physical island is represented by a 1
- the virtual placeholder configuration read (VPCR) object reports the actual module configuration used by the island bus

The Control and Control Status Objects

When the remote virtual placeholder is enabled, 2 special objects in the CANopen fieldbus object dictionary can be used to let the fieldbus master control the island's physical configuration:

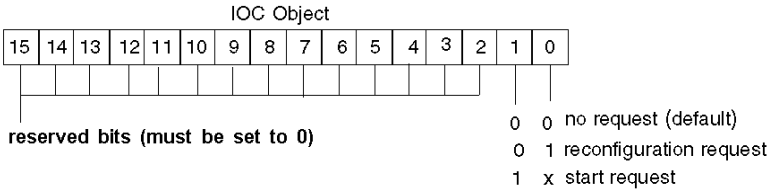
- the IOC object at index 4200 hex
- the IOS object at index 4201 hex

The IOC Object

The IOC object is a read-write 16-bit word. The fieldbus master writes to IOC object only with SDOs, not with PDOs.

The IOC object provides two control functions that enable the fieldbus master to:

- request that a new remote configuration be used on the island
- send a start command to the island



Bit 0 is the reconfiguration request bit. The fieldbus master sets this bit after it has written a new configuration to the VPCW object.

Bit 1 is the start request bit. The fieldbus master sends a start request to the island after the island has successfully processed the reconfiguration request. When the remote virtual placeholder option is enabled, the island requires an explicit start request before it can go into run mode.

NOTE: Writing a new request to the IOC object while the island is in test mode produces an error, and the request is ignored.

The IOS Object

The IOS object is a read-only 16-bit word. It provides status information about the two IOC control functions and displays error codes related to the remote virtual placeholder operation.

Value of the IOS Object	Meaning	Result
0000 hex	Busy	Either no request has been made or a request is being processed but is not complete.
0001 hex	Reconfiguration done	The island has successfully processed a reconfiguration request from the fieldbus master using the value in the VPCW object. The island then waits for a start request.
0002 hex	Start request succeeded	The island has received and processed a start request and can now exchange data with the fieldbus.
0100 hex	Reconfiguration failed	See NIM diagnostics for more details.
0200 hex	Start failed	See NIM diagnostics for more details.
1000 hex	Wrong request	The request is refused.
1100 hex	Non-STB modules marked as not present in VPCW	The request is refused.
1200 hex	Island is being controlled by Advantys configuration software	The request is refused.
The remaining IOS object values are reserved.		

The Write and Write Status Objects

When the remote virtual placeholder is enabled, 2 special objects in the CANopen fieldbus object dictionary can be used to let the fieldbus master write new physical configurations to the island and check the status of the island's configuration:

- the VPCW object at index 4202 hex
- the VPCR object at index 4203 hex

The VPCW Object

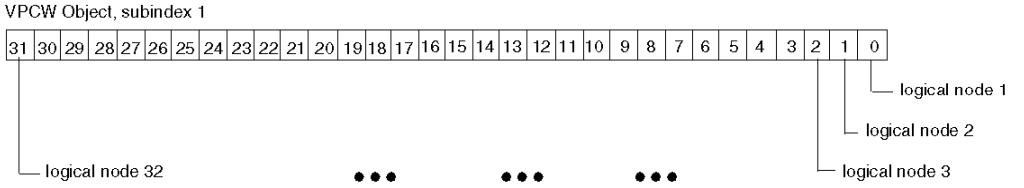
The VPCW object has 3 subindices:

- Subindices 1 and 2 are a pair of 32-bit write-only blocks where the fieldbus master can write a configuration of up to 64 I/O modules on an island bus
- Subindex 0 defines the number of subindices in the object. The value 2 indicates that there are 2 additional subindices beyond subindex 0.

Because the Advantys STBNCO2212 NIM supports a maximum of 32 modules, any values written to subindex 2 are ignored in a remote virtual placeholder operation.

The fieldbus master writes to VPCW object only with SDOs, not with PDOs. The VPCW object is a write-only object. Any attempt to read this object will result in an SDO abort.

Each bit in VPCW subindex 1 represents a logical location on the island bus between address 1 and address 32.

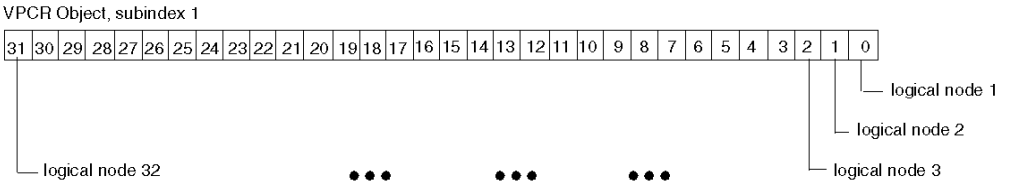


When the fieldbus master writes a 1 to a bit in this object, it configures the logical node associated with that bit as not present in the physical island; i.e., the logical node does not exist on the physical island. A value of 0 in a bit indicates that a module is expected to be present at a specific associated logical node.

For example, if the fieldbus master writes a value of 0 0 0 0 0 0 8 4 hex to this VPCW subindex, then logical nodes 3 and 8 are not expected to be present on the physical island.

The VPCR Object

The VPCR object has the same 3-subindex structure as the VPCW object, with subindex 1 again the most important one. Subindex 1 is a 32-bit block similar to subindex 1 of the VPCW object, where each bit represents a potential logical node on the island bus.



In the VPCR object, the bit pattern in subindex 1 represents the actual configuration being used by the island bus. When the fieldbus master makes a reconfiguration request, it should check this subindex in the VPCR object. After the reconfiguration request is successfully processed, the value in VPCR subindex 1 should be the same as the value in VPCW subindex 1.

Appendices

PL7 Programming Example: a Premium PLC that Supports Remote Virtual Placeholder Operations

A

Overview

The following example describes how to set up an Advantys STB island so that it can run in various I/O re configurations using the remote virtual placeholder option. The fieldbus master is TSXCPP110 CANopen communication module in a Premium PLC.

PL7 is the programming software. Code fragments are included throughout the example to illustrate how the fieldbus master issues SDOs and how the PLC monitors the island's configuration status during the reconfiguration and start processes.

What's in this Chapter?

This chapter contains the following topics:

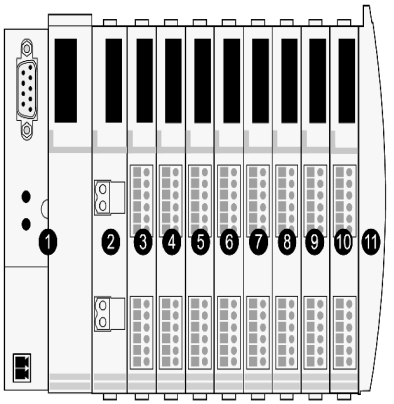
The Remote Virtual Placeholder Operating Environment

Introduction

The following discussion describes the full-option island and the plan of how some of the I/O modules can be removed to support different physical island configurations.

The Full-option Island

The full-option island comprises the NIM, the power distribution module, and all the I/O modules that need to be present to support all desired configurations of the island bus. For our example, we use the STB NCO 2212 CANopen NIM, a 24 VDC PDM, and 8 Advantys STB I/O modules.



1 STB NCO 2212 CANopen NIM (version 3.x or greater)
2 STB PDT 3100 Power Distribution Module
3 STB DDI 3230 two-channel digital input module at island bus logical address 1
4 STB DDO 3200 two-channel digital output module at island bus logical address 2
5 STB DDI 3420 four-channel digital input module at island bus logical address 3
6 STB DDO 3410 four-channel digital output module at island bus logical address 4
7 STB DDI 3610 six-channel digital input module at island bus logical address 5
8 STB DDO 3600 six-channel digital output module at island bus logical address 6
9 STB AVI 1270 2-channel analog input module at island bus logical address 7
10 STB AVO 1250 2-channel analog output module at island bus address 8
11 STB XMP 1100 termination plate

Optional Island Configurations

The island described above is being implemented to support a machine that can be deployed with two optional features. One of the options is controlled by analog I/O channels (option 1). The other option requires two digital input and two digital output channels (option 2). The remaining I/O modules on the island bus are used in all deployments of the machine.

The remote virtual placeholder plan identifies which I/O modules are always present and which may be not present at the island bus addresses, depending on the options you want to use in the machine.

I/O Module	Present in the Island Configuration	Physical Island Address
STB DDI 3230	when option 2 is used	1 when option 2 is used
STB DDO 3200		2 when option 2 is used
STB DDI 3420	always	1 when option 2 is not used
		3 when option 2 is used
STB DDO 3410	always	2 when option 2 is not used
		4 when option 2 is used
STB DDI 3610	always	3 when option 2 is not used
		5 when option 2 is used
STB DDO 3600	always	4 when option 2 is not used
		6 when option 2 is used
STB AVI 1270	when option 1 is used	7 when options 1 and 2 are used
		5 when option 1 is used and option 2 is not used
STB AVO 1250	when option 1 is used	8 when options 1 and 2 are used
		6 when option 1 is used and option 2 is not used

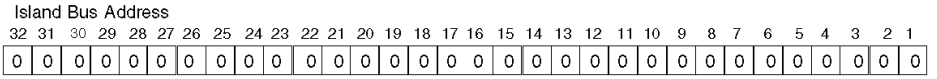
Four configurations of the island are possible:

- where options 1 and 2 are both used (a full-option configuration)
- where option 1 is used and option 2 is not used
- where option 2 is used and option 1 is not used
- where neither option is used

Defining the Configurations as CANopen Objects

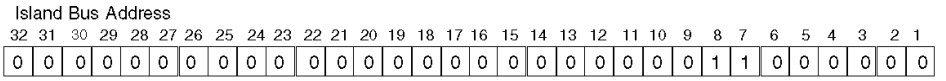
A remote virtual placeholder configuration is represented in the CANopen NIM as a 32-bit object, where each bit represents a logical address on the island bus *(see page 177)*. A bit value of 0 indicates that either a module is expected to be present at that address or no module was configured for that address in the full-option configuration (i.e., in the island's process image). A bit value of 1 indicates that a module that has been defined in the process image is not expected to be present in the physical island configuration.

In a full-option configuration, where the physical island configuration matches the original process image, the object should be:



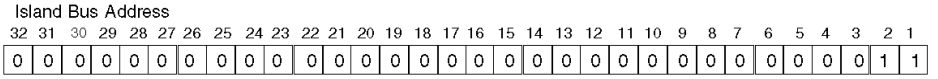
All eight I/O modules should be present in the physical island.

If option 1 is not used in the island configuration, the object should be:



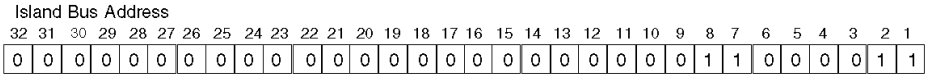
In the physical island, the two analog I/O modules should not be present.

If option 2 is not used in the island configuration, the object should be:



In the physical island, the two 2-channel digital I/O modules should not be present.

If options 1 and 2 are not used in the island configuration, the object should be:



In the physical island, the two analog I/O modules and the two 2-channel digital I/O modules should not be present.

The Application Example

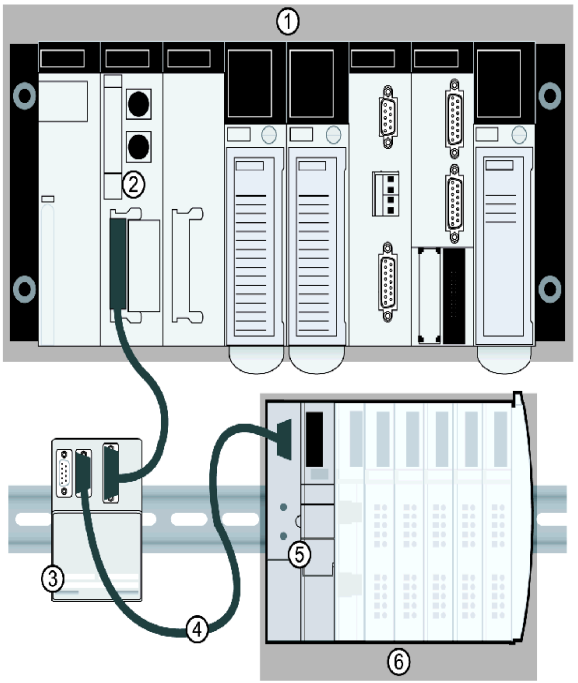
For this example, the island configuration contains option 2 (the 2-channel digital I/O modules at island addresses 1 and 2), and the configuration does not include option 1 (the analog I/O modules are not present at island addresses 7 and 8. The physical island takes the following form:

- STB DDI 3230 input module at address 1
- STB DDO 3200 output module at address 2
- STB DDI 3420 input module at address 3

- STB DDO 3410 output module at address 4
- STB DDI 3610 input module at address 5
- STB DDO 3600 output module at address 6

No other addressable modules are present in this island configuration. When you build the physical island, place only the six I/O modules listed above on the island bus.

The following diagram shows the Premium PLC and STB NCO 2212 NIM communication link over a CANopen network.



1 Premium controller configuration
2 TSX CPP 110 CANopen master PCMCIA card
3 TSX CPP ACC1 CANopen tap junction
4 CANopen network cable (not supplied)
5 STB NCO 2212 CANopen NIM
6 Advantys STB island

NOTE: Notice that the physical island contains only 6 I/O modules because the two analog modules have been removed from the configuration.

A Remote Configuration Example

Summary

The following example describes how to configure the island with the remote virtual placeholder capability and write an optional configuration to the NIM. This example uses an Advantys STB I/O configuration where option 2 is included and option 1 is not present *(see page 186)*.

Constructing the Physical Island

You need to construct a physical island that contains all the modules that need to be present in the desired configuration and does not contain any of the modules that are not scheduled to be present. For this example, six I/O modules need to be present:

- an STB DDI 3230 input module at island address 1
- an STB DDO 3200 output module at island address 2
- an STB DDI3420 input module at island address 3
- an STB DDO 3410 output module at island address 4
- an STB DDI 3610 input module at island address 5
- an STB DDO 3600 output module at island address 6

Configuring the Full-option System with the Advantys Configuration Software

The island must be initially configured with the full-option system *(see page 184)* and the NIM must be configured to support the remote virtual placeholder option. This configuration contains all the I/O modules, including the option 1 and option 2 I/O modules. The initial configuration requires the Advantys configuration software.

Step	Action
1	With the module editor, configure the fieldbus handler control word in the NIM to support remote virtual placeholder *(see page 128)*.
2	With the module editor in the software, set the desired operating parameters for all the I/O modules.
3	Export an EDS file *(see page 64)* from the Advantys configuration software to the CANopen fieldbus master and use this file to complete the CANopen master configuration *(see page 116)*.
4	Connect and download the complete configuration to the NIM.

PLC Memory Variables for Remote Virtual Placeholder Operations

At this point, you have an island with a physical configuration that does not match the full-option configuration downloaded to the NIM. An island with the remote virtual placeholder capability implemented does not automatically go into run mode at power-up. Several actions must first be taken by the PLC in order to run the island with a valid configuration.

First, you need to set up some memory variables in the Premium PLC to support the remote virtual placeholder operations. For this example, the memory variables of interest are as follows:

Memory Variable	Content	Value
%MW298	The island's node ID on the CANopen network.	7 for this example
%MW300	Exchange number	Managed by the system
%MW301	Communication status	Managed by the system
%MW302	Timeout value in units of 10 ms	Managed by the user
%MW303	Number of bytes to send for WRITE_VAR	Managed by the user
	Number of bytes received for READ_VAR	Managed by the system
%MW305	IOC object	
%MW306	IOS object	
%MW310	Modules 1 ... 16 in the VPCW object	C0 hex
%MW311	Modules 17 ... 32 in the VPCW object	00 hex
%MW312	Modules 33 ... 48 in the VPCW object	00 hex
%MW313	Modules 49 ... 64 in the VPCW object	00 hex
%MW315	Modules 1 ... 16 in the VPCR object	
%MW316	Modules 17 ... 32 in the VPCR object	
%MW317	Modules 33 ... 48 in the VPCR object	
%MW318	Modules 49 ... 64 in the VPCR object	

NOTE: The memory variables %MW300 ... %MW303 are required parameters for the PL7 software to issue READ_VAR and WRITE_VAR commands.

NOTE: The fieldbus master will configure the island with option 1 not present. The value in memory address %MW310 is C0 hex, indicating that the module configured for island addresses 7 and 8 in the full-option configuration are not present in the configuration that will be sent by the fieldbus master.

Checking the IOS

Before the fieldbus master can write a new virtual placeholder configuration to the island, the PLC needs to check the IOS object in the NIM to make sure it is set to 0001 hex. A value of 1 indicates that the NIM has a configuration and is ready to run. Using PL7, issue an SDO read as follows:

```
(* Check IOS *)
%MW302:=100; (* SDO timeout = 100 x 10ms *)
READ_VAR(ADR#0.1.SYS,'SDO',16#00004201,%MW298,%MW306:1,
%MW300:4);
```

The IOS object (%MW306) should contain a value of 1 because the NIM has a configuration (the full-option configuration).

Writing the Remote Virtual Placeholder Configuration to the VPCW Object]

The next step is for the PLC to write the new remote virtual placeholder configuration to subindex 1of the VPCW object. The request needs to be sent with an SDO write.

```
(* Send SDO Upload Request to VPCW - for modules 1-32 *)
%MW302:=100; (* SDO timeout = 100 x 10 ms *)
%MW303:=4; (* Number of bytes to write *)
WRITE_VAR(ADR#0.1.SYS,'SDO',16#00014202,%MW298,%MW310:2,
%MW300:4);
```

The VPCW object now contains the new configuration for 6 I/O modules instead of 8, with the 2 option 1 analog modules not present.

Making the Reconfiguration Request

The PLC now must send an SDO with a reconfiguration request to the NIM. This request will cause the NIM to write the configuration in the VPCW object to its Flash.

```
(* Send Request to Reconfigure the Island *)
%MW302:=100; (* SDO timeout = 100 x 10 ms *)
%MW303:=2;(* Number of bytes to write *)
%MW305=1; (* IOC - Reconfigure *)
WRITE_VAR(ADR#0.1.SYS,'SDO',16#00004200,%MW298,%MW305:1,
%MW300:4);
```

After the new configuration is written to Flash, the NIM reboots the island bus and sets the value in the IOS object to 0001hex. This status value indicates that the island has a configuration again (in this case, the new one written by the fieldbus master) and is ready to run.

Comparing the VPCW and the VPCR

The value in the VPCR object indicates the true I/O configuration of the physical island. The PLC should send an SDO query to the NIM to make sure that the VPCR object matches the desired remote virtual placeholder configuration.

```
(* Query Actual Virtual Placeholder Conf. of the Island *)
%MW302:=100; (* SDO timeout = 100 x 10 ms *)
READ_VAR(ADR#0.1.SYS,'SDO',16#00014203,%MW298,%MW315:2,
%MW300:4);
```

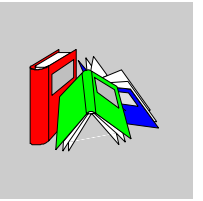
%MW315 should contain the actual remote virtual placeholder configuration used in the island. If the values of the two objects do not match, the start request will not succeed.

Making the Start Request

After the PLC confirms that the new configuration in Flash matches the actual physical configuration, it can issue a start request.

```
(* Send Start Request to the Island *)
%MW302:=100; (* SDO timeout = 100 x 10 ms *)
%MW303:=2;(* Number of bytes to write *)
%MW305:=2; (* IOC - Start *)
WRITE_VAR(ADR#0.1.SYS,'SDO',16#00004200,%MW298,%MW305:1,
%MW300:4);
```

Glossary



0-9

100Base-T

An adaptation of the IEEE 802.3u (Ethernet) standard, the 100Base-T standard uses twisted-pair wiring with a maximum segment length of 100 m (328 ft) and terminates with an RJ-45 connector. A 100Base-T network is a baseband network capable of transmitting data at a maximum speed of 100 Mbit/s. "Fast Ethernet" is another name for 100Base-T, because it is ten times faster than 10Base-T.

10Base-T

An adaptation of the IEEE 802.3 (Ethernet) standard, the 10Base-T standard uses twisted-pair wiring with a maximum segment length of 100 m (328 ft) and terminates with an RJ-45 connector. A 10Base-T network is a baseband network capable of transmitting data at a maximum speed of 10 Mbit/s.

802.3 frame

A frame format, specified in the IEEE 802.3 (Ethernet) standard, in which the header specifies the data packet length.

A

agent

1. SNMP – the SNMP application that runs on a network device.

2. Fipio – a slave device on a network.

analog input

A module that contains circuits that convert analog DC input signals to digital values that can be manipulated by the processor. By implication, these analog inputs are usually direct. That means a data table value directly reflects the analog signal value.

analog output

A module that contains circuits that transmit an analog DC signal proportional to a digital value input to the module from the processor. By implication, these analog outputs are usually direct. That means a data table value directly controls the analog signal value.

application object

In CAN-based networks, application objects represent device-specific functionality, such as the state of input or output data.

ARP

The ARP (address resolution protocol) is the IP network layer protocol, which uses ARP to map an IP address to a MAC (hardware) address.

auto baud

The automatic assignment and detection of a common baud rate as well as the ability of a device on a network to adapt to that rate.

auto-addressing

The assignment of an address to each Island bus I/O module and preferred device.

auto-configuration

The ability of Island modules to operate with predefined default parameters. A configuration of the Island bus based completely on the actual assembly of I/O modules.

B

basic I/O

Low-cost Advantys STB input/output modules that use a fixed set of operating parameters. A basic I/O module cannot be reconfigured with the Advantys Configuration Software and cannot be used in reflex actions.

basic network interface

A low-cost Advantys STB network interface module that supports up to 12 Advantys STB I/O modules. A basic NIM does not support the Advantys Configuration Software, reflex actions, nor the use of an HMI panel.

basic power distribution module

A low-cost Advantys STB PDM that distributes sensor power and actuator power over a single field power bus on the Island. The bus provides a maximum of 4 A total power. A basic PDM requires a 5 A fuse to protect the I/O.

BootP

BootP (bootstrap protocol) is an UDP/IP protocol that allows an internet node to obtain its IP parameters based on its MAC address.

BOS

BOS stands for beginning of segment. When more than 1 segment of I/O modules is used in an Island, an STB XBE 1200 or an STB XBE 1300 BOS module is installed in the first position in each extension segment. Its job is to carry Island bus communications to and generate logic power for the modules in the extension segment. Which BOS module must be selected depends on the module types that shall follow.

bus arbitrator

A master on a Fipio network.

C

CAN

The CAN (controller area network) protocol (ISO 11898) for serial bus networks is designed for the interconnection of smart devices (from multiple manufacturers) in smart systems for real-time industrial applications. CAN multi-master systems ensure high data integrity through the implementation of broadcast messaging and advanced diagnostic mechanisms. Originally developed for use in automobiles, CAN is now used in a variety of industrial automation control environments.

CANopen protocol

An open industry standard protocol used on the internal communication bus. The protocol allows the connection of any enhanced CANopen device to the Island bus.

CI

This abbreviation stands for command interface.

CiA

CiA (CAN in Automation) is a non-profit group of manufacturers and users dedicated to developing and supporting CAN-based higher layer protocols.

CIP

Common Industrial Protocol. Networks that include CIP in the application layer can communicate seamlessly with other CIP-based networks. For example, the implementation of CIP in the application layer of an Ethernet TCP/IP network creates an EtherNet/IP environment. Similarly, CIP in the application layer of a CAN network creates a DeviceNet environment. Devices on an EtherNet/IP network can therefore communicate with devices on a DeviceNet network via CIP bridges or routers.

COB

A COB (communication object) is a unit of transportation (a message) in a CAN-based network. Communication objects indicate a particular functionality in a device. They are specified in the CANopen communication profile.

configuration

The arrangement and interconnection of hardware components within a system and the hardware and software selections that determine the operating characteristics of the system.

CRC

cyclic redundancy check. Messages that implement this error checking mechanism have a CRC field that is calculated by the transmitter according to the message's content. Receiving nodes recalculate the field. Disagreement in the two codes indicates a difference between the transmitted message and the one received.

CSMA/CS

carrier sense multiple access/collision detection. CSMA/CS is a MAC protocol that networks use to manage transmissions. The absence of a carrier (transmission signal) indicates that a network channel is idle. Multiple nodes may try to simultaneously transmit on the channel, which creates a collision of signals. Each node detects the collision and immediately terminates transmission. Messages from each node are retransmitted at random intervals until the frames are successfully transmitted.

D

DDXML

Device Description eXtensible Markup Language

device name

A customer-driven, unique logical personal identifier for an Ethernet NIM. A device name (or *role name*) is created when you combine the numeric rotary switch setting with the NIM (for example, STBNIP2212_010).

After the NIM is configured with a valid device name, the DHCP server uses it to identify the island at power up.

DeviceNet protocol

DeviceNet is a low-level, connection-based network that is based on CAN, a serial bus system without a defined application layer. DeviceNet, therefore, defines a layer for the industrial application of CAN.

DHCP

dynamic host configuration protocol. A TCP/IP protocol that allows a server to assign an IP address based on a device name (host name) to a network node.

differential input

A type of input design where two wires (+ and -) are run from each signal source to the data acquisition interface. The voltage between the input and the interface ground are measured by two high-impedance amplifiers, and the outputs from the two amplifiers are subtracted by a third amplifier to yield the difference between the + and - inputs. Voltage common to both wires is thereby removed. Differential design solves the problem of ground differences found in single-ended connections, and it also reduces the cross-channel noise problem.

digital I/O

An input or output that has an individual circuit connection at the module corresponding directly to a data table bit or word that stores the value of the signal at that I/O circuit. It allows the control logic to have discrete access to the I/O values.

DIN

Deutsche industrial norms. A German agency that sets engineering and dimensional standards and now has worldwide recognition.

Drivecom Profile

The Drivecom profile is part of CiA DSP 402 (profile), which defines the behavior of drives and motion control devices on CANopen networks.

E

economy segment

A special type of STB I/O segment created when an STB NCO 1113 economy CANopen NIM is used in the first location. In this implementation, the NIM acts as a simple gateway between the I/O modules in the segment and a CANopen master. Each I/O module in an economy segment acts as a independent node on the CANopen network. An economy segment cannot be extended to other STB I/O segments, preferred modules or enhanced CANopen devices.

EDS

electronic data sheet. The EDS is a standardized ASCII file that contains information about a network device's communications functionality and the contents of its object dictionary. The EDS also defines device-specific and manufacturer-specific objects.

EIA

Electronic Industries Association. An organization that establishes electrical/electronic and data communication standards.

EMC

electromagnetic compatibility. Devices that meet EMC requirements can operate within a system's expected electromagnetic limits without interruption.

EMI

electromagnetic interference. EMI can cause an interruption, malfunction, or disturbance in the performance of electronic equipment. It occurs when a source electronically transmits a signal that interferes with other equipment.

EOS

This abbreviation stands for end of segment. When more than 1 segment of I/O modules is used in an Island, an STB XBE 1000 or an STB XBE 1100 EOS module is installed in the last position in every segment that has an extension following it. The EOS module extends Island bus communications to the next segment. Which EOS module must be selected depends on the module types that shall follow.

Ethernet

A LAN cabling and signaling specification used to connect devices within a defined area, e.g., a building. Ethernet uses a bus or a star topology to connect different nodes on a network.

Ethernet II

A frame format in which the header specifies the packet type, Ethernet II is the default frame format for NIM communications.

EtherNet/IP

EtherNet/IP (the Ethernet Industrial Protocol) is especially suited to factory applications in which there is a need to control, configure, and monitor events within an industrial system. The ODVA-specified protocol runs CIP (the Common Industrial Protocol) on top of standard Internet protocols, like TCP/IP and UDP. It is an open local (communications) network that enables the interconnectivity of all levels of manufacturing operations from the plant's office to the sensors and actuators on its floor.

F

fallback state

A known state to which an Advantys STB I/O module can return in the event that its communication connection is not open.

fallback value

The value that a device assumes during fallback. Typically, the fallback value is either configurable or the last stored value for the device.

FED_P

Fipio extended device profile. On a Fipio network, the standard device profile type for agents whose data length is more than 8 words and equal to or less than 32 words.

Fipio

Fieldbus Interface Protocol (FIP). An open fieldbus standard and protocol that conforms to the FIP/World FIP standard. Fipio is designed to provide low-level configuration, parameterization, data exchange, and diagnostic services.

Flash memory

Flash memory is nonvolatile memory that can be overwritten. It is stored on a special EEPROM that can be erased and reprogrammed.

FRD_P

Fipio reduced device profile. On a Fipio network, the standard device profile type for agents whose data length is two words or less.

FSD_P

Fipio standard device profile. On a Fipio network, the standard device profile type for agents whose data length is more than two words and equal to or less than 8 words.

full scale

The maximum level in a specific range—e.g., in an analog input circuit the maximum allowable voltage or current level is at full scale when any increase beyond that level is over-range.

function block

A function block performs a specific automation function, such as speed control. A function block comprises configuration data and a set of operating parameters.

function code

A function code is an instruction set commanding 1 or more slave devices at a specified address(es) to perform a type of action, e.g., read a set of data registers and respond with the content.

G

gateway

A program or hardware that passes data between networks.

global_ID

global_identifier. A 16-bit integer that uniquely identifies a device's location on a network. A global_ID is a symbolic address that is universally recognized by all other devices on the network.

GSD

generic slave data (file). A device description file, supplied by the device's manufacturer, that defines a device's functionality on a Profibus DP network.

H

HMI

human-machine interface. An operator interface, usually graphical, for industrial equipment.

hot swapping

Replacing a component with a like component while the system remains operational. When the replacement component is installed, it begins to function automatically.

HTTP

hypertext transfer protocol. The protocol that a web server and a client browser use to communicate with one another.

I

I/O base

A mounting device, designed to seat an Advantys STB I/O module, hang it on a DIN rail, and connect it to the Island bus. It provides the connection point where the module can receive either 24 VDC or 115/230 VAC from the input or output power bus distributed by a PDM.

I/O module

In a programmable controller system, an I/O module interfaces directly to the sensors and actuators of the machine/process. This module is the component that mounts in an I/O base and provides electrical connections between the controller and the field devices. Normal I/O module capacities are offered in a variety of signal levels and capacities.

I/O scanning

The continuous polling of the Advantys STB I/O modules performed by the COMS to collect data bits, status, nd diagnostics information.

IEC

International Electrotechnical Commission Carrier. Founded in 1884 to focus on advancing the theory and practice of electrical, electronics, and computer engineering, and computer science. EN 61131-2 is the specification that deals with industrial automation equipment.

IEC type 1 input

Type 1 digital inputs support sensor signals from mechanical switching devices such as relay contacts and push buttons operating in normal environmental conditions.

IEC type 2 input

Type 2 digital inputs support sensor signals from solid state devices or mechanical contact switching devices such as relay contacts, push buttons (in normal or harsh environmental conditions), and 2- or 3-wire proximity switches.

IEC type 3 input

Type 3 digital inputs support sensor signals from mechanical switching devices such as relay contacts, push buttons (in normal-to-moderate environmental conditions), 3-wire proximity switches and 2-wire proximity switches that have:

- a voltage drop of no more than 8 V
- a minimum operating current capability less than or equal to 2.5 mA
- a maximum off-state current less than or equal to 1.5 mA

IEEE

Institute of Electrical and Electronics Engineers, Inc. The international standards and conformity assessment body for all fields of electrotechnology, including electricity and electronics.

industrial I/O

An Advantys STB I/O module designed at a moderate cost for typical continuous, high-duty-cycle applications. Modules of this type often feature standard IEC threshold ratings, usually providing user-configurable parameter options, on-board protection, good resolution, and field wiring options. They are designed to operate in moderate-to-high temperature ranges.

input filtering

The amount of time that a sensor must hold its signal on or off before the input module detects the change of state.

input polarity

An input channel's polarity determines when the input module sends a 1 and when it sends a 0 to the master controller. If the polarity is *normal*, an input channel sends a 1 to the controller when its field sensor turns on. If the polarity is *reverse*, an input channel sends a 0 to the controller when its field sensor turns on.

input response time

The time it takes for an input channel to receive a signal from the field sensor and put it on the Island bus.

INTERBUS protocol

The INTERBUS fieldbus protocol observes a master/slave network model with an active ring topology, having all devices integrated in a closed transmission path.

IOC object

Island operation control object. A special object that appears in the CANopen object dictionary when the remote virtual placeholder option is enabled in a CANopen NIM. It is a 16-bit word that provides the fieldbus master with a mechanism for issuing reconfiguration and start requests.

IOS object

Island operation status object. A special object that appears in the CANopen object dictionary when the remote virtual placeholder option is enabled in a CANopen NIM. It is a 16-bit word that reports the success of reconfiguration and start requests or records diagnostic information in the event that a request is not completed.

IP

internet protocol. That part of the TCP/IP protocol family that tracks the internet addresses of nodes, routes outgoing messages, and recognizes incoming messages.

IP Rating

Ingress Protection rating according to IEC 60529.

IP20 modules are protected against ingress and contact of objects larger than 12.5 mm. The module is not protected against harmful ingress of water.

IP67 modules are completely protected against ingress of dust and contact. Ingress of water in harmful quantity is not possible when the enclosure is immersed in water up to 1 m.

L

LAN

local area network. A short-distance data communications network.

light industrial I/O

An Advantys STB I/O module designed at a low cost for less rigorous (e.g., intermittent, low-duty-cycle) operating environments. Modules of this type operate in lower temperature ranges with lower qualification and agency requirements and limited on-board protection; they usually have limited or no user-configuration options.

linearity

A measure of how closely a characteristic follows a straight-line function.

LSB

least significant bit, least significant byte. The part of a number, address, or field that is written as the rightmost single value in conventional hexadecimal or binary notation.

M

MAC address

media access control address. A 48-bit number, unique on a network, that is programmed into each network card or device when it is manufactured.

mandatory module

When an Advantys STB I/O module is configured to be mandatory, it must be present and healthy in the Island configuration for the Island to be operational. If a mandatory module is inoperable or is removed from its location on the Island bus, the Island goes to a pre-operational state. By default, all I/O modules are not mandatory. You must use the Advantys Configuration Software to set this parameter.

master/slave model

The direction of control in a network that implements the master/slave model is always from the master to the slave devices.

Modbus

Modbus is an application layer messaging protocol. Modbus provides client and server communications between devices connected on different types of buses or networks. Modbus offers many services specified by function codes.

MOV

metal oxide varistor. A 2-electrode semiconductor device with a voltage-dependant nonlinear resistance that drops markedly as the applied voltage is increased. It is used to suppress transient voltage surges.

MSB

most significant bit, most significant byte. The part of a number, address, or field that is written as the leftmost single value in conventional hexadecimal or binary notation.

N

N.C. contact

normally closed contact. A relay contact pair that is closed when the relay coil is de-energized and open when the coil is energized.

N.O. contact

normally open contact. A relay contact pair that is open when the relay coil is de-energized and closed when the coil is energized.

NEMA

National Electrical Manufacturers Association

network cycle time

The time that a master requires to complete a single scan of all of the configured I/O modules on a network device; typically expressed in microseconds.

NIM

network interface module. This module is the interface between an Island bus and the fieldbus network of which the Island is a part. A NIM enables all the I/O on the Island to be treated as a single node on the fieldbus. The NIM also provides 5 V of logic power to the Advantys STB I/O modules in the same segment as the NIM.

NMT

network management. NMT protocols provide services for network initialization, diagnostic control, and device status control.

O

object dictionary

Part of the CANopen device model that provides a map to the internal structure of CANopen devices (according to CANopen profile DS-401). A device's object dictionary (also called the *object directory*) is a lookup table that describes the data types, communications objects, and application objects the device uses. By accessing a particular device's object dictionary through the CANopen fieldbus, you can predict its network behavior and build a distributed application.

ODVA

Open Devicenet Vendors Association. The ODVA supports the family of network technologies that are built on the Common Industrial Protocol (EtherNet/IP, DeviceNet, and CompoNet).

open industrial communication network

A distributed communication network for industrial environments based on open standards (EN 50235, EN50254, and EN50170, and others) that allows the exchange of data between devices from different manufacturers.

output filtering

The amount that it takes an output channel to send change-of-state information to an actuator after the output module has received updated data from the NIM.

output polarity

An output channel's polarity determines when the output module turns its field actuator on and when it turns the actuator off. If the polarity is *normal*, an output channel turns its actuator on when the master controller sends it a 1. If the polarity is *reverse*, an output channel turns its actuator on when the master controller sends it a 0.

output response time

The time it takes for an output module to take an output signal from the Island bus and send it to its field actuator.

P

parameterize

To supply the required value for an attribute of a device at run-time.

PDM

power distribution module. A module that distributes either AC or DC field power to a cluster of I/O modules directly to its right on the Island bus. A PDM delivers field power to the input modules and the output modules. It is important that all the I/O clustered directly to the right of a PDM be in the same voltage group—either 24 VDC, 115 VAC, or 230 VAC.

PDO

process data object. In CAN-based networks, PDOs are transmitted as unconfirmed broadcast messages or sent from a producer device to a consumer device. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

PE

protective earth. A return line across the bus for fault currents generated at a sensor or actuator device in the control system.

peer-to-peer communications

In peer-to-peer communications, there is no master/slave or client/server relationship. Messages are exchanged between entities of comparable or equivalent levels of functionality, without having to go through a third party (like a master device).

PLC

programmable logic controller. The PLC is the brain of an industrial manufacturing process. It automates a process as opposed to relay control systems. PLCs are computers suited to survive the harsh conditions of the industrial environment.

PowerSuite Software

PowerSuite Software is a tool for configuring and monitoring control devices for electric motors, including ATV31, ATV71, and TeSys U.

preferred module

An I/O module that functions as an auto-addressable device on an Advantys STB Island but is not in the same form factor as a standard Advantys STB I/O module and therefore does not fit in an I/O base. A preferred device connects to the Island bus via an EOS module and a length of a preferred module extension cable. It can be extended to another preferred module or back into a BOS module. If it is the last device on the Island, it must be terminated with a 120 Ω terminator.

premium network interface

A premium NIM has advanced features over a standard or basic NIM.

prioritization

An optional feature on a standard NIM that allows you to selectively identify digital input modules to be scanned more frequently during a the NIM's logic scan.

process I/O

An Advantys STB I/O module designed for operation at extended temperature ranges in conformance with IEC type 2 thresholds. Modules of this type often feature high levels of on-board diagnostics, high resolution, user-configurable parameter options, and higher levels of agency approval.

process image

A part of the NIM firmware that serves as a real-time data area for the data exchange process. The process image includes an input buffer that contains current data and status information from the Island bus and an output buffer that contains the current outputs for the Island bus, from the fieldbus master.

producer/consumer model

In networks that observe the producer/consumer model, data packets are identified according to their data content rather than by their node address. All nodes *listen* on the network and consume those data packets that have appropriate identifiers.

Profibus DP

Profibus Decentralized Peripheral. An open bus system that uses an electrical network based on a shielded 2-wire line or an optical network based on a fiber-optic cable. DP transmission allows for high-speed, cyclic exchange of data between the controller CPU and the distributed I/O devices.

R

reflex action

A simple, logical command function configured locally on an Island bus I/O module. Reflex actions are executed by Island bus modules on data from various Island locations, like input and output modules or the NIM. Examples of reflex actions include compare and copy operations.

repeater

An interconnection device that extends the permissible length of a bus.

reverse polarity protection

Use of a diode in a circuit to protect against damage and unintended operation in the event that the polarity of the applied power is accidentally reversed.

rms

root mean square. The effective value of an alternating current, corresponding to the DC value that produces the same heating effect. The rms value is computed as the square root of the average of the squares of the instantaneous amplitude for 1 complete cycle. For a sine wave, the rms value is 0.707 times the peak value.

role name

A customer-driven, unique logical personal identifier for an Ethernet NIM. A role name (or *device name*) is created when you:

- combine the numeric rotary switch setting with the NIM (for example, STBNIP2212_010), or . . .
- edit the **Device Name** setting in the NIM's embedded web server pages

After the NIM is configured with a valid role name, the DHCP server uses it to identify the island at power up.

RTD

resistive temperature detect. An RTD device is a temperature transducer composed of conductive wire elements typically made of platinum, nickel, copper, or nickel-iron. An RTD device provides a variable resistance across a specified temperature range.

RTP

run-time parameters. RTP lets you monitor and modify selected I/O parameters and Island bus status registers of the NIM while the Advantys STB Island is running. The RTP feature uses 5 reserved output words in the NIM's process image (the RTP request block) to send requests, and 4 reserved input words in the NIM's process image (the RTP response block) to receive responses. Available only in standard NIMs running firmware version 2.0 or higher.

Rx

reception. For example, in a CAN-based network, a PDO is described as an RxPDO of the device that receives it.

S

SAP

service access point. The point at which the services of 1 communications layer, as defined by the ISO OSI reference model, is made available to the next layer.

SCADA

supervisory control and data acquisition. Typically accomplished in industrial settings by means of microcomputers.

SDO

service data object. In CAN-based networks, SDO messages are used by the fieldbus master to access (read/write) the object directories of network nodes.

segment

A group of interconnected I/O and power modules on an Island bus. An Island must have at least 1 segment and, depending on the type of NIM used, may have as many as 7 segments. The first (leftmost) module in a segment needs to provide logic power and Island bus communications to the I/O modules on its right. In the primary or basic segment, that function is filled by a NIM. In an extension segment, that function is filled by an STB XBE 1200 or an STB XBE 1300 BOS module.

SELV

safety extra low voltage. A secondary circuit designed and protected so that the voltage between any 2 accessible parts (or between 1 accessible part and the PE terminal for Class 1 equipment) does not exceed a specified value under normal conditions or under single-fault conditions.

SIM

subscriber identification module. Originally intended for authenticating users of mobile communications, SIMs now have multiple applications. In Advantys STB, configuration data created or modified with the Advantys Configuration Software can be stored on a SIM (referred to as the “removable memory card”) and then written to the NIM’s Flash memory.

single-ended inputs

An analog input design technique whereby a wire from each signal source is connected to the data acquisition interface, and the difference between the signal and ground is measured. For the success of this design technique, 2 conditions are imperative: the signal source must be grounded, and the signal ground and data acquisition interface ground (the PDM lead) must have the same potential.

sink load

An output that, when turned on, receives DC current from its load.

size 1 base

A mounting device, designed to seat an STB module, hang it on a DIN rail, and connect it to the Island bus. It is 13.9 mm (0.55 in.) wide and 128.25 mm (5.05 in.) high.

size 2 base

A mounting device, designed to seat an STB module, hang it on a DIN rail, and connect it to the Island bus. It is 18.4 mm (0.73 in.) wide and 128.25 mm (5.05 in.) high.

size 3 base

A mounting device, designed to seat an STB module, hang it on a DIN rail, and connect it to the Island bus. It is 28.1 mm (1.11 in.) wide and 128.25 mm (5.05 in.) high.

slice I/O

An I/O module design that combines a small number of channels (usually between 2 and 6) in a small package. The idea is to allow a system developer to purchase just the right amount of I/O and to be able to distribute it around the machine in an efficient, mechatronics way.

SM_MPS

state management_message periodic services. The applications and network management services used for process control, data exchange, diagnostic message reporting, and device status notification on a Fipio network.

SNMP

simple network management protocol. The UDP/IP standard protocol used to manage nodes on an IP network.

snubber

A circuit generally used to suppress inductive loads—it consists of a resistor in series with a capacitor (in the case of an RC snubber) and/or a metal-oxide varistor placed across the AC load.

source load

A load with a current directed into its input; must be driven by a current source.

standard I/O

Any of a subset of Advantys STB input/output modules designed at a moderate cost to operate with user-configurable parameters. A standard I/O module may be reconfigured with the Advantys Configuration Software and, in most cases, may be used in reflex actions.

standard network interface

An Advantys STB network interface module designed at moderate cost to support the configuration capabilities, multi-segment design and throughput capacity suitable for most standard applications on the Island bus. An Island run by a standard NIM can support up to 32 addressable Advantys STB and/or preferred I/O modules, up to 12 of which may be standard CANopen devices.

standard power distribution module

An Advantys STB module that distributes sensor power to the input modules and actuator power to the output modules over two separate power buses on the Island. The bus provides a maximum of 4 A to the input modules and 8 A to the output modules. A standard PDM requires a 5 A fuse to protect the input modules and an 8 A fuse to protect the outputs.

STD_P

standard profile. On a Fipio network, a standard profile is a fixed set of configuration and operating parameters for an agent device, based on the number of modules that the device contains and the device's total data length. There are 3 types of standard profiles: Fipio reduced device profile (FRD_P), Fipio standard device profile (FSD_P), and the Fipio extended device profile (FED_P).

stepper motor

A specialized DC motor that allows discrete positioning without feedback.

subnet

A part of a network that shares a network address with the other parts of a network. A subnet may be physically and/or logically independent of the rest of the network. A part of an internet address called a subnet number, which is ignored in IP routing, distinguishes the subnet.

surge suppression

The process of absorbing and clipping voltage transients on an incoming AC line or control circuit. Metal-oxide varistors and specially designed RC networks are frequently used as surge suppression mechanisms.

T

TC

thermocouple. A TC device is a bimetallic temperature transducer that provides a temperature value by measuring the voltage differential caused by joining together two different metals at different temperatures.

TCP

transmission control protocol. A connection-oriented transport layer protocol that provides reliable full-duplex data transmission. TCP is part of the TCP/IP suite of protocols.

telegram

A data packet used in serial communication.

TFE

transparent factory Ethernet. Schneider Electric's open automation framework based on TCP/IP.

Tx

transmission. For example, in a CAN-based network, a PDO is described as a TxPDO of the device that transmits it.

U

UDP

user datagram protocol. A connectionless mode protocol in which messages are delivered in a datagram to a destination computer. The UDP protocol is typically bundled with the Internet Protocol (UPD/IP).

V

varistor

A 2-electrode semiconductor device with a voltage-dependant nonlinear resistance that drops markedly as the applied voltage is increased. It is used to suppress transient voltage surges.

voltage group

A grouping of Advantys STB I/O modules, all with the same voltage requirement, installed directly to the right of the appropriate power distribution module (PDM) and separated from modules with different voltage requirements. Never mix modules with different voltage requirements in the same voltage group.

VPCR object

virtual placeholder configuration read object. A special object that appears in the CANopen object dictionary when the remote virtual placeholder option is enabled in a CANopen NIM. It provides a 32-bit subindex that represents the actual module configuration used in a physical Island.

VPCW object

virtual placeholder configuration write object. A special object that appears in the CANopen object dictionary when the remote virtual placeholder option is enabled in a CANopen NIM. It provides a 32-bit subindex where the fieldbus master can write a module reconfiguration. After the fieldbus writes to the VPCW subindex, it can issue a reconfiguration request to the NIM that begins the remote virtual placeholder operation.

W

watchdog timer

A timer that monitors a cyclical process and is cleared at the conclusion of each cycle. If the watchdog runs past its programmed time period, it generates a fault.

Index

D

E

F

G

H

I

L

M

N

O

P

R

S

T

V