CANopen Router/B

User Manual

A-CANOR/B

Document No. D150-007
01/2026
Revision 1.11

aparian

CONTENTS

Revision History

Revision	Date	Comment
1.0	2 July 2021	Initial document
1.1	1 Feb 2022	Added Comms Lost Action for EIP Originator IO devices. Added note regarding CAN interface inhibit when a Ethernet Comms Originator.
1.2	9 Mar 2022	Added CANopen implementation specification
1.3	13 Mar 2022	Added ODVA and UL (C1D2) Conformance Mark.
1.4	4 Aug 2022	Added information required for UL regarding open type device enclosures.
1.5	17 Jan 2023	Update support contact details
1.6	11 Sep 2023	Add Master Node for when the module is sending Heartbeats as a CANopen Master. Added an additional SDO passthrough CIP message that is 32-bit aligned. Added option for CANopen using extended framing.
1.7	8 November 2023	Added ATEX Conformance Mark Added UKCA Conformance Mark
1.8	28 November 2023	Updated EtherNet/IP Class 1 connection import options
1.9	20 May 2025	Add duplicate IP address indication to LEDs in the Diagnostics section.
1.10	26 August 2025	Added Virtual Slave Map Item Transaction Count to the Logix Assembly and Modbus Mapping (when operating as a CANopen Slave). Added Virtual Slaves page in the Diagnostics.
1.11	28 January 2026	Added section on the Internal Register Viewer. Add network parameter upload to SD Card section. Add new RS232 and RS485 statistics.

1. PREFACE

1.1. INTRODUCTION TO THE CANOPEN ROUTER

This manual describes the installation, operation, and diagnostics of the Aparian CANopen Router Series B module. The CANopen Router/B, (hereafter referred to as the **module**) provides intelligent data routing between either EtherNet/IP or Modbus TCP/RTU and the CANopen bus network. This allows the user to integrate CANopen devices into a Rockwell Logix platform (e.g., ControlLogix or CompactLogix) or any Modbus Master or Slave device with minimal effort.

The module can be configured to be either a CANopen Master or CANopen Slave allowing the user to not only integrate CANopen devices into a Logix or Modbus system, but to also allow the user to use EtherNet/IP or Modbus devices in an existing CANopen network (by using the CANopen Router/B in Slave mode).

CANopen Master
When the module operates as a CANopen Master, it can connect to a maximum of 124 CANopen Slaves. The process data (PDOs) from each CANopen Slave can be mapped to any of the operating interfaces (EtherNet/IP Target, Modbus Slave, Modbus Master, or EtherNet/IP Originator).

CANopen Slave
When the module is configured to be a CANopen Slave, it can emulate up to 128 Process Data Objects (PDOs) with each mapped item having the ability to be configured as a separate CANopen node.

The module can use one of four interface modes:

EtherNet/IP Target
As a EtherNet/IP target, the module can use one of two methods to read and write data to and from the CANopen network:

- **Direct-To-Tag technology**

  This allows the CANopen Master or Slaves to exchange data with a Logix controller without the need to write any ladder or application code in Studio 5000. The CANopen data is directly read from, or written to, Logix tags.

- **EtherNet/IP Class 1 connection**

  Here a remote EtherNet/IP device (e.g. a Logix controller) establishes a number of Class 1 connections to the module. CANopen data can be mapped into two separate input and output class 1 cyclic connections to the Logix controller (allowing up to 1KB input and 1KB output to be exchanged at the requested packet interval – RPI).

Modbus Slave
The diagnostics and CANopen data (from either CANopen Master or Slaves) will be written to, or read from, the module's internal Modbus Registers (Holding or Input Registers). These registers can be accessed by a remote Modbus Master using either Modbus TCP, Modbus RTU232, or Modbus RTU485.

Modbus Master
The diagnostics and CANopen data (from either CANopen Master or Slaves) will be written to, or read from, the module's internal Modbus Registers (Holding or Input Registers). The Modbus Auxiliary Map can then be used to configure the Modbus data exchange between multiple remote Modbus Slave devices and the module's internal Modbus registers. The Modbus communication can be via Modbus TCP, Modbus RTU232, or Modbus RTU485.

EtherNet/IP Originator
As an EtherNet/IP originator, the module can use one of two methods to read and write data to and from the CANopen network:

- **EtherNet/IP Explicit Messaging**

  This allows the CANopen Master or Slaves to exchange data with up to 5 EtherNet/IP devices. The module can use either Class 3 or Unconnected Messaging (UCMM) to Get and Set data in the remote EtherNet/IP devices.

- **EtherNet/IP Class 1 connection**

  CANopen data (from either CANopen Master or Slaves) can be mapped to a max of 5 EtherNet/IP devices using input and output class 1 cyclic connections. This will allow the CANopen Router/B to "own" the EtherNet/IP target device and exchange CANopen data using the EtherNet/IP device's input and output assemblies.

The CANopen Router/B can map up to 100 Service Data Objects (SDOs) from various CANopen Slaves into any of the of the operating interfaces (EtherNet/IP Target, Modbus Slave, Modbus Master, or EtherNet/IP Originator) similar to the mapping of CANopen Slave Process Data Objects (PDOs). Additionally, the SDO map supports and option to write a static value to an SDO on start-up.

The CANopen Router/B is configured using the Aparian Slate application. This program can be downloaded from www.aparian.com free of charge.

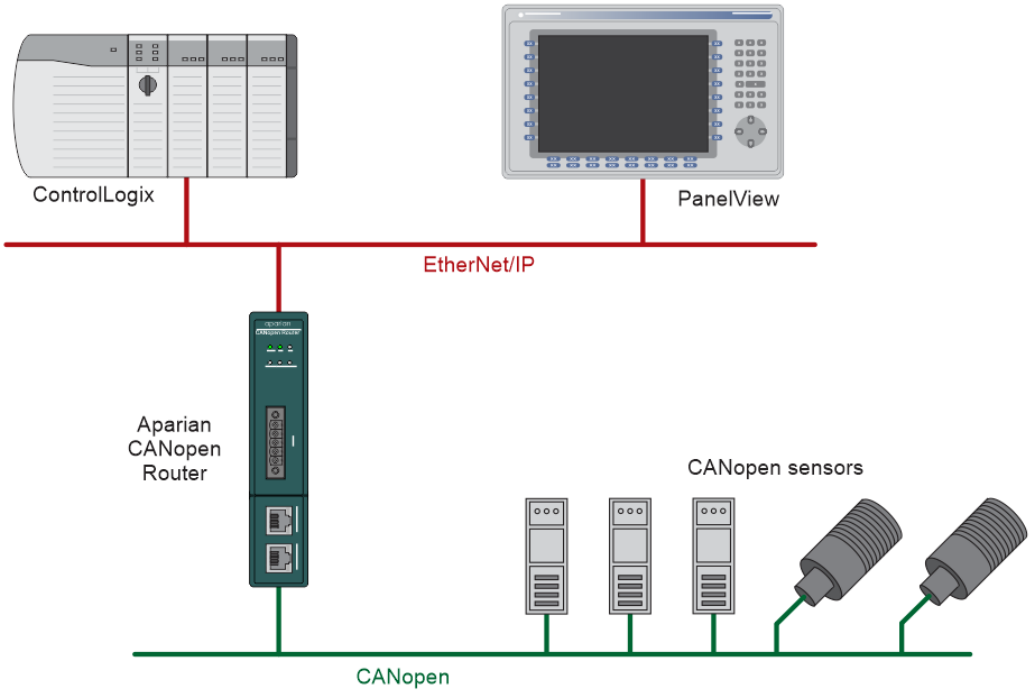


Figure 1.1. – Typical CANopen Master architecture using the CANopen Router/B

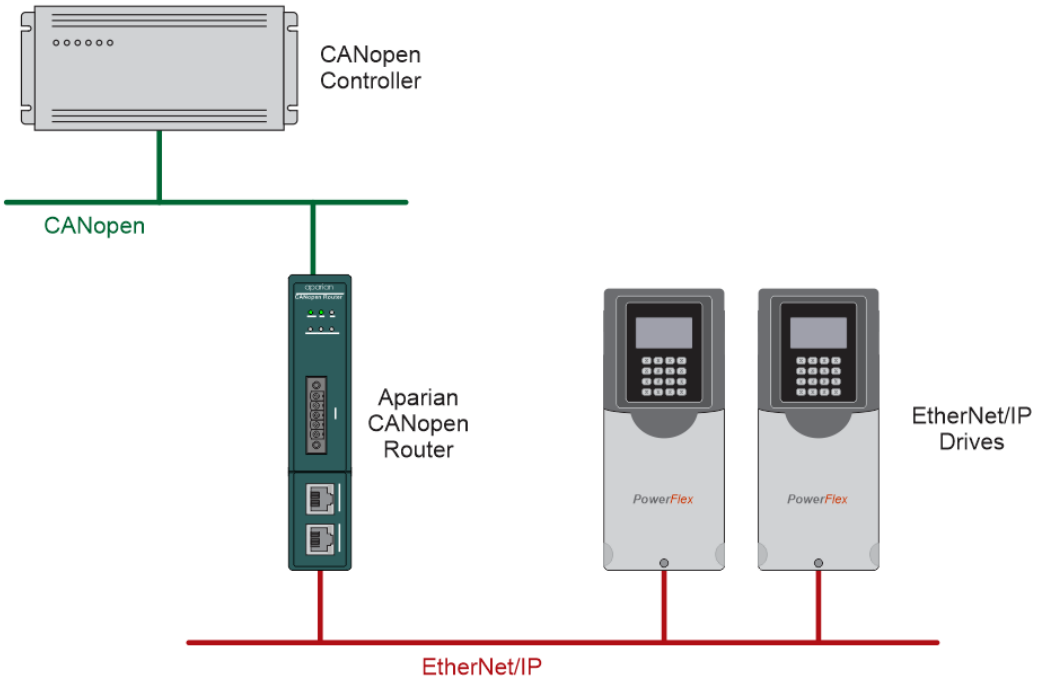


Figure 1.2. – Typical CANopen Slave architecture using the CANopen Router

When operating as a CANopen Master, Slate allows the user to map up to 32 PDOs per CANopen Slave to any of the operating interfaces (EtherNet/IP Target, Modbus Slave, Modbus Master, or EtherNet/IP Originator). The module and Slate will allow the user to parameterize

each CANopen Slave according to the parameters provided in the CANopen Slave EDS file. These parameters can be saved in the CANopen Slave Device Non-Volatile memory.

The module also provides a range of statistics, CANopen and Modbus packet capture, and internal Modbus and Data table reads to simplify the diagnostic process for remote diagnosis.

The module has two Ethernet ports and supports Device-Level-Ring (DLR) architectures.

A built-in webserver provides detailed diagnostics of system configuration and operation, including the display of CANopen operation and communication statistics, without the need for any additional software.

1.2. FEATURES

- Module can operate as a CANopen Master or Slave.
- CANopen Master mode can configure and operate with up to 124 CANopen Slaves.
- CANopen Slave mode can emulate up to 128 PDOs with various CANopen node addresses.
- Module has various operating interfaces:
    - EtherNet/IP Target (Class 1 connection as well as Direct-To-Tag Logix tag access)
    - Modbus Slave (TCP, RTU232, and RTU485)
    - Modbus Master (TCP, RTU232, and RTU485)
    - EtherNet/IP Originator (Class 1 connection with up to 5 EtherNet/IP devices and Explicit Messaging with up to 5 EtherNet/IP devices).
- Support for up to 32 PDOs (receive and transmit) per CANopen Slave.
- Support for mapping of up to 128 SDOs to any of the operating interfaces (also supports writing of static value once off write).
- Slate software provides a CANopen and Modbus packet capture for better diagnosis of issues.
- Supports all CANopen Baud Rates (10k, 20k, 50k, 125k, 250k, 500k, 800k, 1M).
- Supports CiA 443 Bootloader Auto-enable.
- In Master Mode supports NMT message to initialize network.
- Supports CANopen LSS Node and Bit Rate assignment.
- Time Synchronization of the CANopen network.
- Master supports SYNC for PDO communication.
- Supports all error and emergency (EMCY) messages and handling.
- Dual Ethernet ports which support Device-Level-Ring (DLR).
- Network Time Protocol (NTP) supported for external time synchronization.

- Small form factor – DIN rail mounted.

1.3. ARCHITECTURE

The figures below provide an example of the typical network setup for connecting CANopen (as a Master or Slave) to EtherNet/IP or Modbus TCP/RTU232/RTU485.

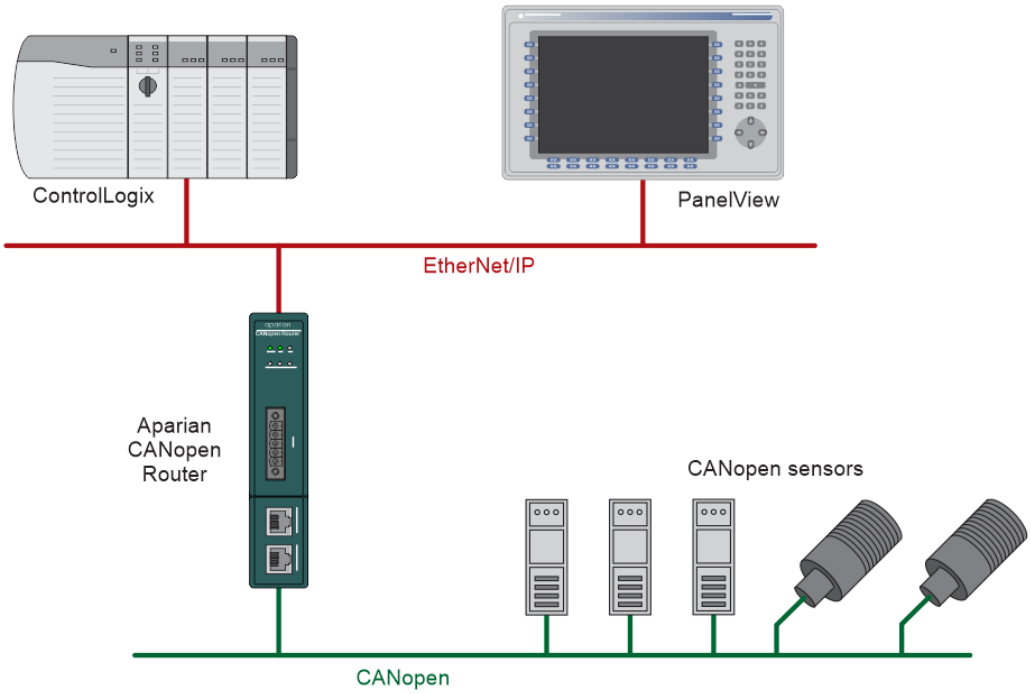


Figure 1.3. – Example of connecting CANopen Slaves to a Logix Controller

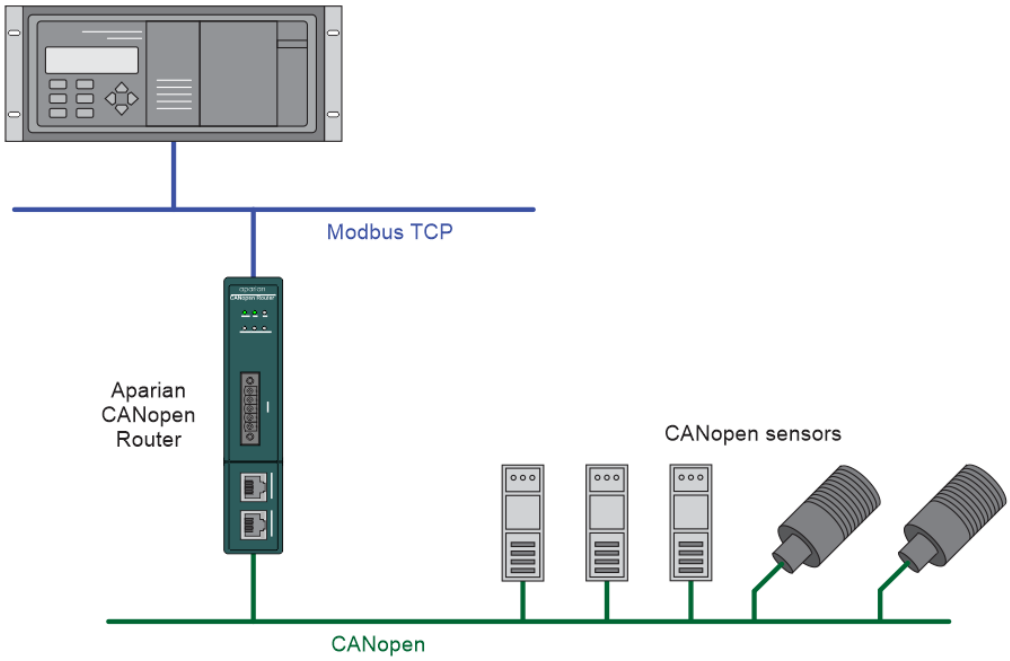


Figure 1.4. - Example of connecting CANopen Slaves to a Modbus TCP Master or Slave

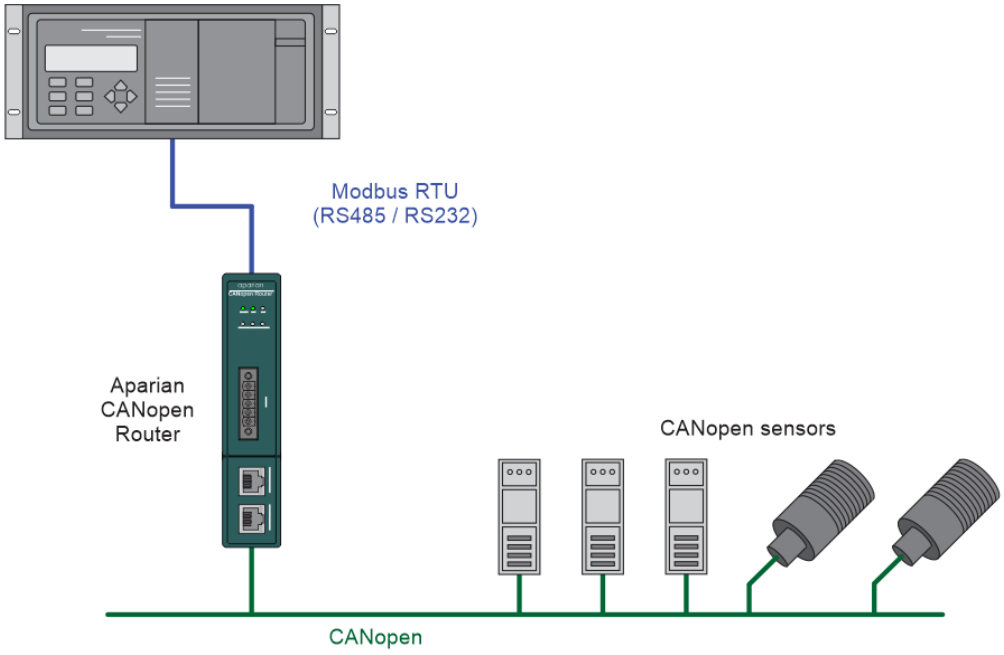


Figure 1.5. - Example of connecting CANopen Slaves to a Modbus RTU Master or Slave

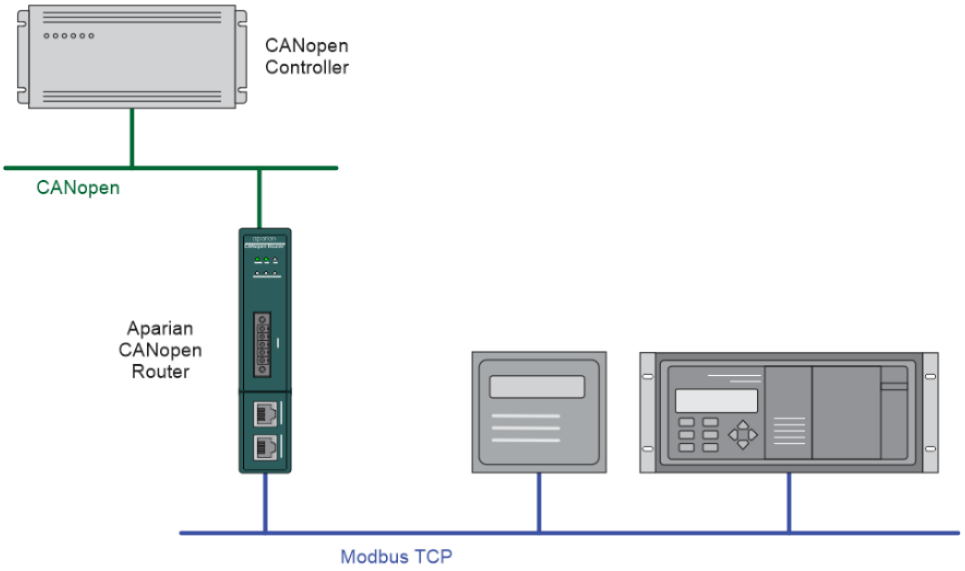


Figure 1.6. – Modbus TCP Device (Master or Slave) operating as a CANopen Slave

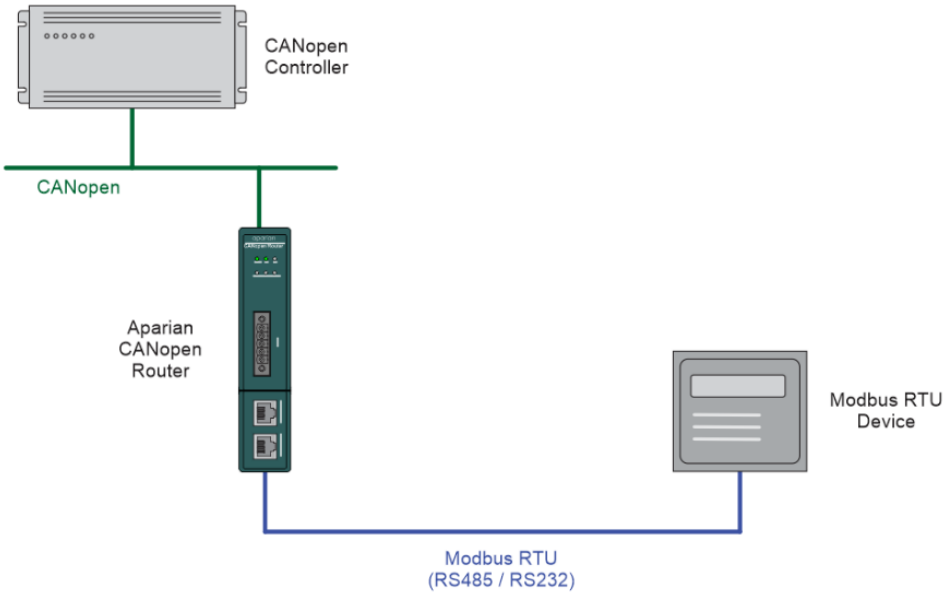


Figure 1.7. – Modbus RTU Device (Master or Slave) operating as a CANopen Slave

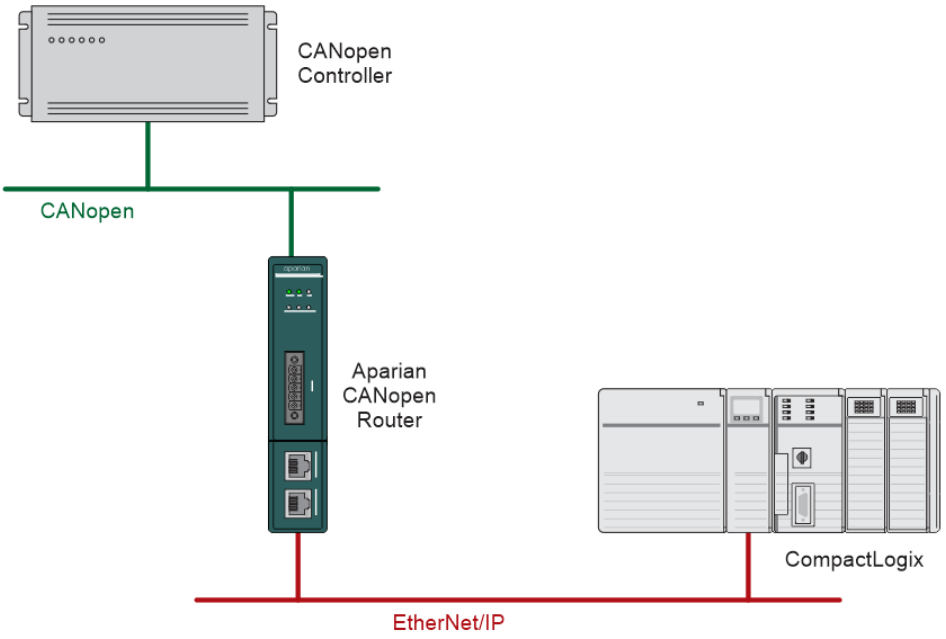


Figure 1.8. – Logix Controller operating as a CANopen Slave via the CANopen Router

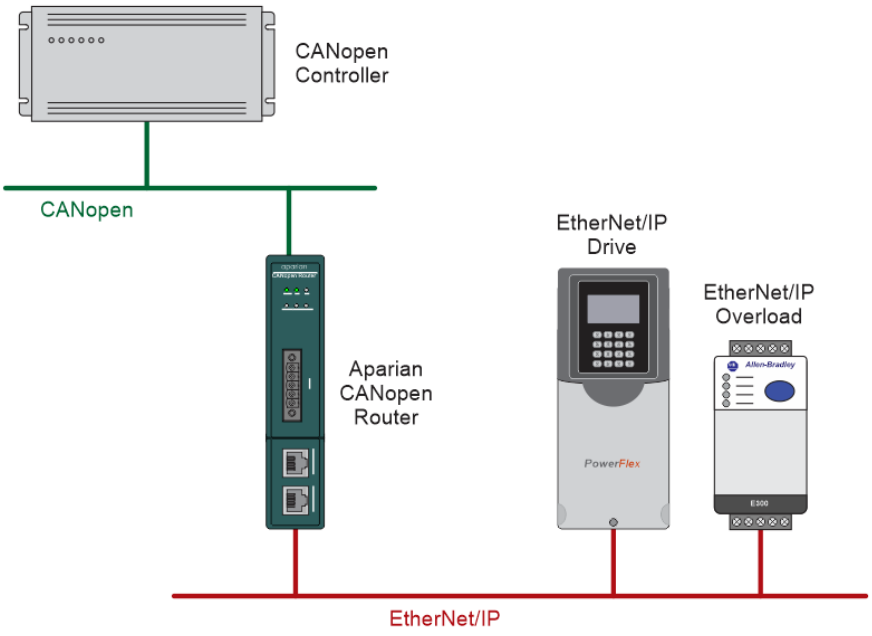


Figure 1.9. – EtherNet/IP Drive and Smart Overload operating as a CANopen Slave

1.4. ADDITIONAL INFORMATION

The following documents contain additional information that can assist the user with the module installation and operation.

Resource	Link
Slate Installation	http://www.aparian.com/software/slate
CANopen Router/B User Manual CANopen Router/B Datasheet Example Code & UDTs	http://www.aparian.com/products/canopenrouterb
Ethernet wiring standard	www.cisco.com/c/en/us/td/docs/video/cds/cde/cde205_220_420/installation/guide/cde205_220_420_hig/Connectors.html
CANopen Standards	https://www.can-cia.org/canopen/

Table 1.1. - Additional Information

1.5. SUPPORT

Technical support is provided via the Web (in the form of user manuals, FAQ, datasheets etc.) to assist with installation, operation, and diagnostics.

For additional support the user can use either of the following:

Resource	Link
Contact Us web link	https://www.prosoft-technology.com/Services-Support/Customer-Support
Support email	support@prosoft-technology.com

Table 1.2. – Support Details

2. INSTALLATION

2.1. MODULE LAYOUT

The module has two ports at the bottom of the enclosure, two Ethernet ports on the angled front, and one port at the front as shown in the figure below. The ports at the bottom are used for RS232 and RS485 serial communication, and power. The power port uses a three-way connector which is used for the DC power supply positive and negative (ground) voltage as well as the earth connection.

The port on the front of the module is the CAN port and can also be used to power the module.

NOTE: The module allows the user to provide power on both bottom and front power connectors and can be used for power supply redundancy.

The Ethernet cable used for the Ethernet ports must be wired according to industry standards which can be found in the additional information section of this document.

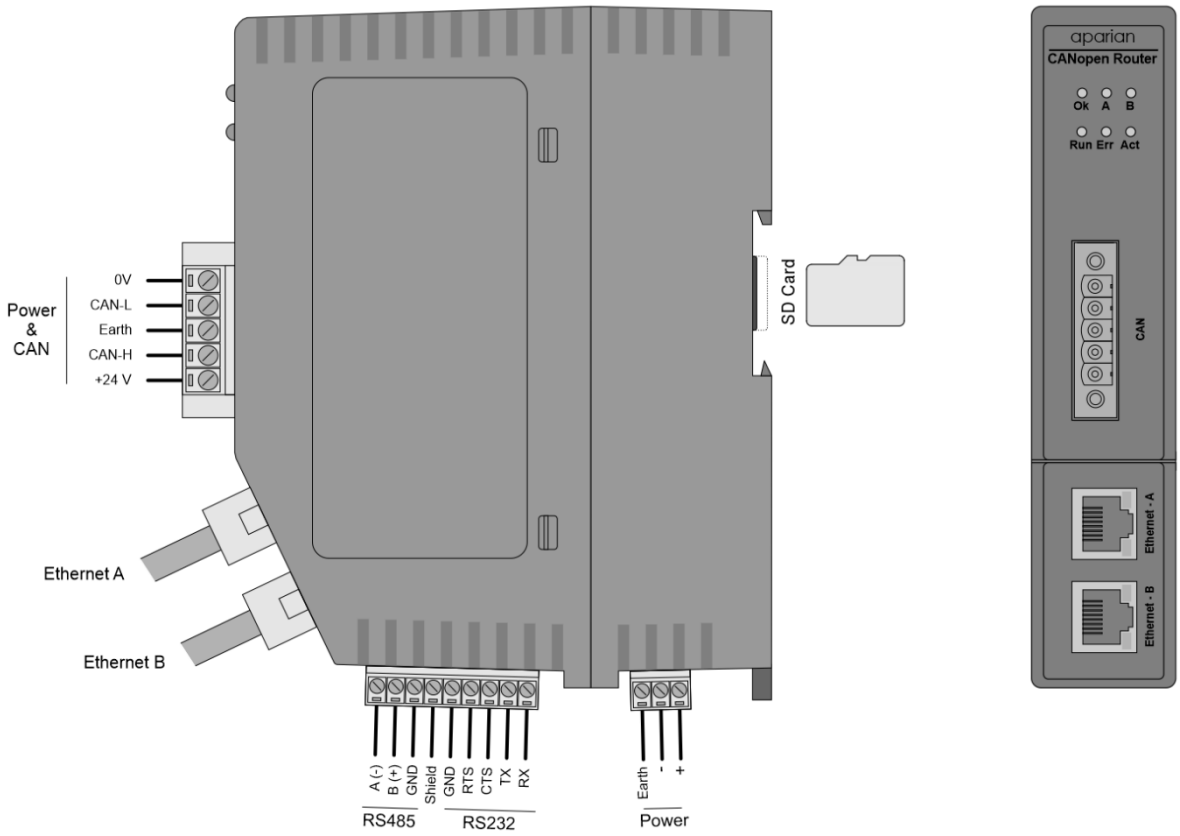


Figure 2.1 – CANopen Router/B side and front view

The module also supports a SD Card for disaster recovery which can be used to automatically update the configuration and/or firmware of a new module.

The module provides six diagnostic LEDs as shown in the front view figure above. These LEDs are used to provide information regarding the module system operation, the Ethernet interface, and the auxiliary communication interface (RS232 or RS485).

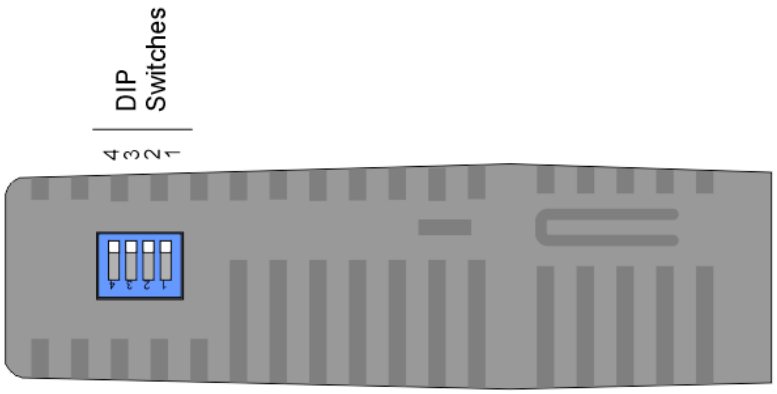


Figure 2.2 – CANopen Router/B top view

The module provides four DIP switches at the top of the enclosure as shown in the top view figure above.

DIP Switch	Description
DIP Switch 1	Used to force the module into “Safe Mode”. When in “Safe Mode” the module will not load the application firmware and will wait for new firmware to be downloaded. This should only be used in the rare occasion when an earlier firmware update was interrupted at a critical stage.
DIP Switch 2	This will force the module into DHCP mode which is useful when the user has forgotten the IP address of the module.
DIP Switch 3	This DIP Switch is used to lock the configuration from being overwritten by the Slate. When set Slate will not be able to download to the module.
DIP Switch 4	When this DIP Switch is set at bootup it will force the module Ethernet IP address to **192.168.1.100** and network mask 255.255.255.0. The user can then switch the DIP switch off and assign the module a static IP address if needed.

Table 2.1 - DIP Switch Settings

2.1. MODULE MOUNTING

NOTE: This module is an open-type device and is meant to be installed in an enclosure suitable for the environment such that the equipment is only accessible with the use of a tool.

The module provides a DIN rail clip to mount onto a 35mm DIN rail.

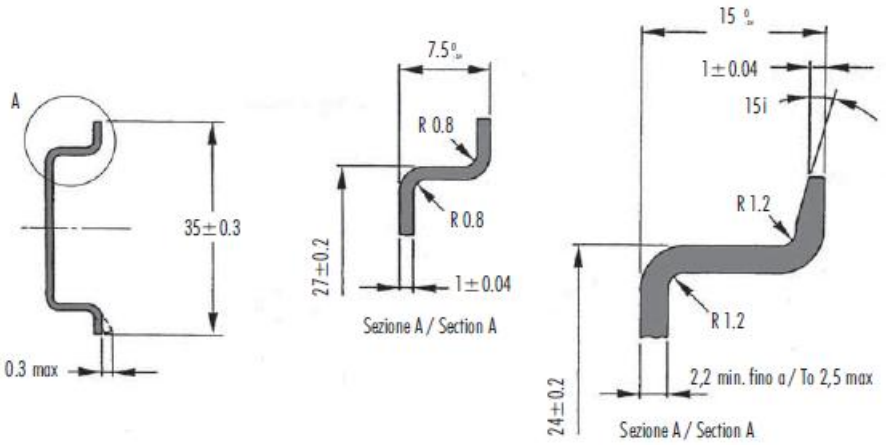


Figure 2.3 - DIN rail specification

The DIN rail clip is mounted on the bottom of the module at the back as shown in the figure below. Use a flat screwdriver to pull the clip downward. This will enable the user to mount the module onto the DIN rail. Once the module is mounted onto the DIN rail the clip must be pushed upwards to lock the module onto the DIN rail.

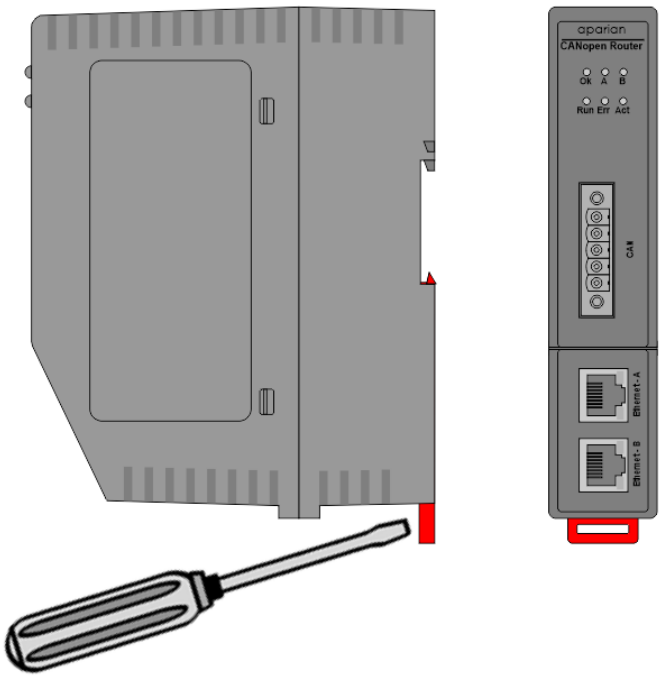


Figure 2.4 - DIN rail mouting

2.2. BOTTOM POWER

A three-way power connector is used to connect Power+, Power– (GND), and earth. The module requires an input voltage of 10 – 32Vdc. **Refer** to the technical specifications section in this document.

NOTE: The module allows the user to provide power on both bottom and front power connectors and can be used for power supply redundancy.

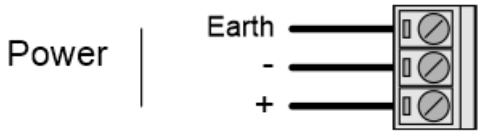


Figure 2.5 - Power connector

2.3. RS232/RS485 PORT

The nine-way connector is used to connect the RS232 and RS485 conductors for serial communication. The shield terminal can be used for shielded cable in high noise environments.

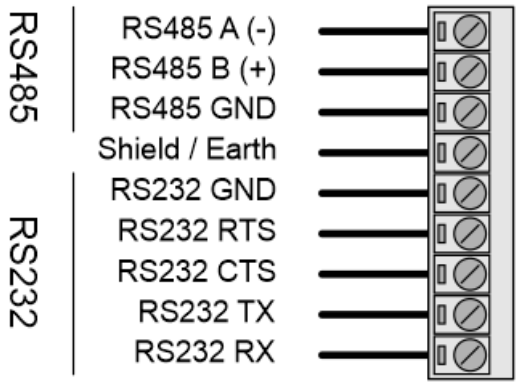


Figure 2.6 - RS232/RS485 connector

The RS485 port provides the standard A and B conductors. The RS232 port provides the standard communication conductors (RX, TX, and GND) as well as hardware handshaking lines for legacy systems (RTS – Request to Send, CTS – Clear to Send).

NOTE: The shield of the RS232/RS485 port is internally connected to the power connector earth. Thus, when using a shield, it is important to connect the Earth terminal on the power connector to a clean earth. Failing to do this can lower the signal quality of the RS232/RS485 communication.

NOTE: When using a shielded cable, it is important that only one end of the shield is connected to earth to avoid current loops. It is recommended to connect the shield to the CANopen Router module, and not to the other CANopen device.

2.4. RS485 TERMINATION

All RS485 networks need to be terminated at the extremities (start and end point) of the communication conductor. The termination for the RS485 network can be enabled/disabled via the module configuration. Enabling the termination will connect an internal 125 Ohm resistor between the positive (B+) and negative (A-) conductors of the RS485 network.

2.5. ETHERNET PORTS

The Ethernet connectors should be wired according to industry standards. **Refer** to the additional information section in this document for further details.

The module has an embedded switch connecting the two Ethernet ports.

2.6. CAN AND FRONT POWER

A five-way CAN connector is used to connect the CANopen CAN bus network as well as the Power+, Power– (GND), and earth. The module requires an input voltage of 10 – 32Vdc. **Refer** to the technical specifications section in this document.

NOTE: The module allows the user to provide power on both bottom and front power connectors and can be used for power redundancy.

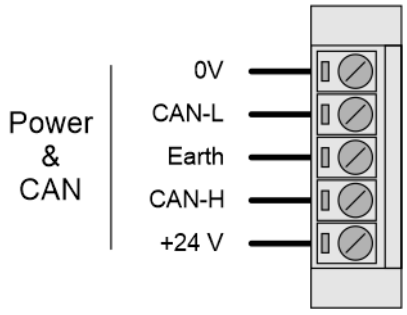


Figure 2.7 – CAN and Power connector

3. SETUP

3.1. INSTALL CONFIGURATION SOFTWARE

All the network setup and configuration of the module is achieved by means of the Aparian Slate device configuration environment. This software can be downloaded from http://www.aparian.com/software/slate.

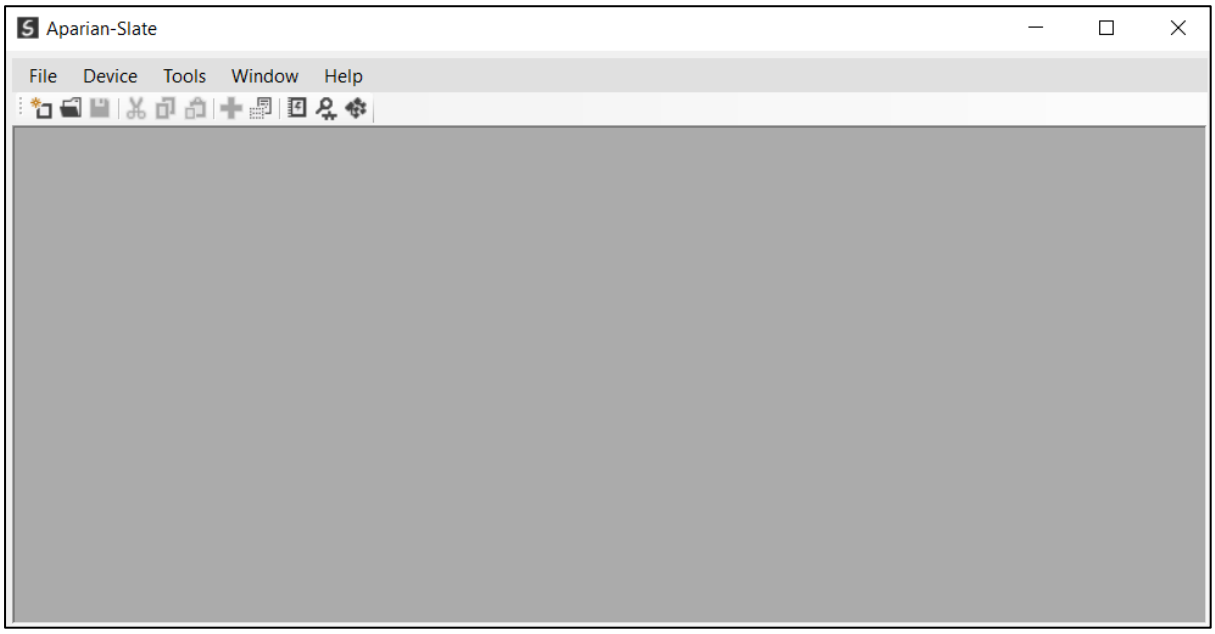


Figure 3.1. - Aparian Slate Environment

3.2. NETWORK PARAMETERS

The module will have DHCP (Dynamic Host Configuration Protocol) enabled as factory default. Thus, a DHCP server must be used to provide the module with the required network parameters (IP address, subnet mask, etc.). There are a number of DHCP utilities available, however it is recommended that the DHCP server in Slate be used.

Within the Slate environment, the DHCP server can be found under the Tools menu.

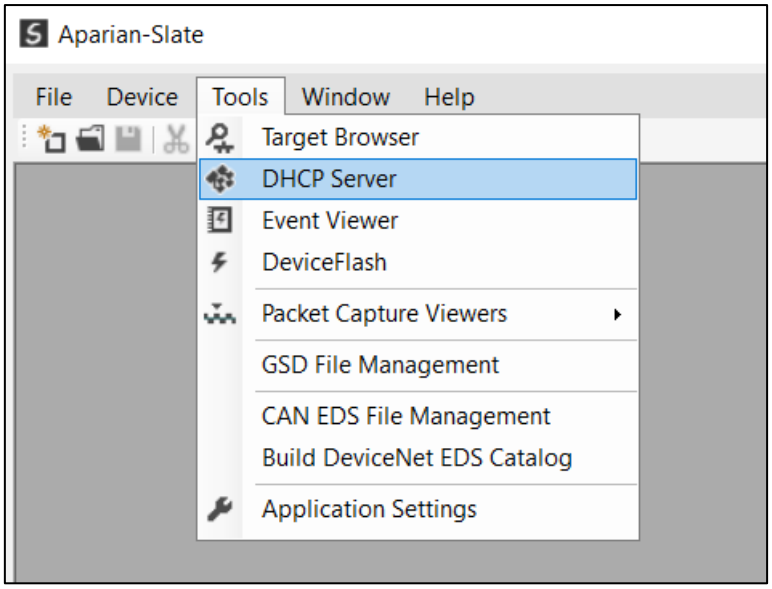


Figure 3.2. - Selecting DHCP Server

Once opened, the DHCP server will listen on all available network adapters for DHCP requests and display their corresponding MAC addresses.

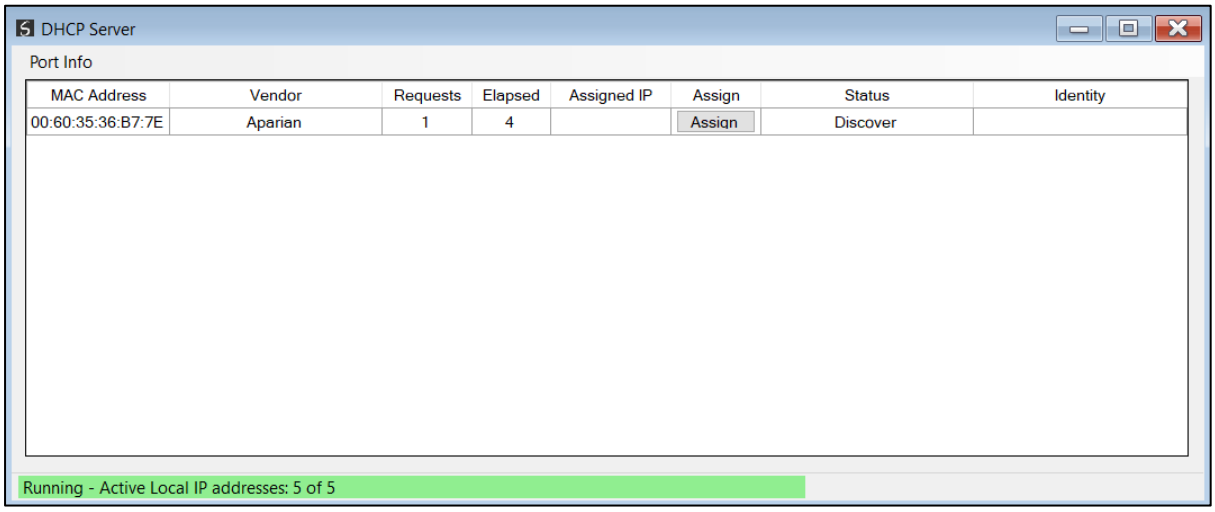


Figure 3.3. - DHCP Server

NOTE: If the DHCP requests are not displayed in the DHCP Server it may be due to the local PC's firewall. During installation, the necessary firewall rules are automatically created for the Windows firewall. Another possibility is that another DHCP Server is operational on the network and it has assigned the IP address.

To assign an IP address, click on the corresponding "Assign" button. The IP Address Assignment window will open.

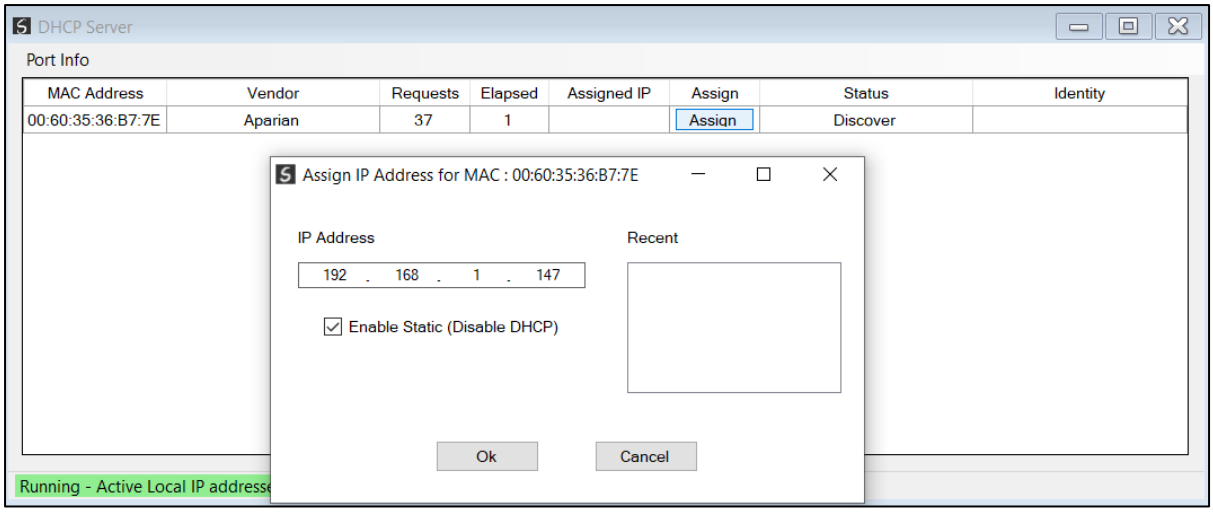


Figure 3.4. - Assigning IP Address

The required IP address can then be either entered, or a recently used IP address can be selected by clicking on an item in the Recent List.

If the “Enable Static” checkbox is checked, then the IP address will be set to static after the IP assignment, thereby disabling future DHCP requests.

Once the IP address window has been accepted, the DHCP server will automatically assign the IP address to the module and then read the Identity object Product name from the device.

The successful assignment of the IP address by the device is indicated by the green background of the associated row.

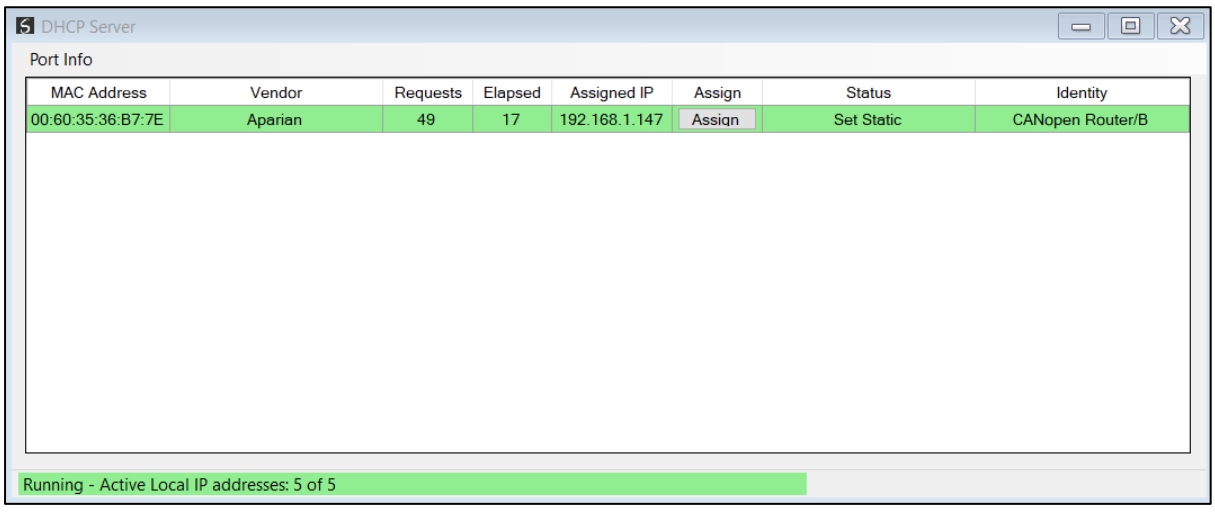


Figure 3.5. - Successful IP address assignment

It is possible to force the module back into DHCP mode by powering up the device with DIP switch 2 set to the On position.

A new IP address can then be assigned by repeating the previous steps.

NOTE: It is important to return DIP switch 2 back to Off position, to avoid the module returning to a DHCP mode after the power is cycled again.

If the module's DIP switch 2 is in the On position during the address assignment, the user will be warned by the following message.

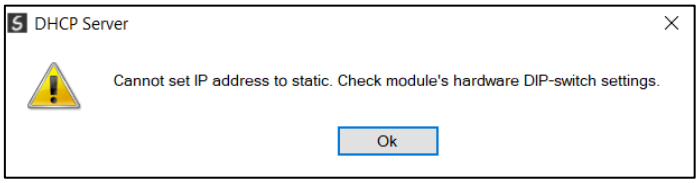


Figure 3.6. - Force DHCP warning

In addition to the setting the IP address, a number of other network parameters can be set during the DHCP process. These settings can be viewed and edited in Slate's Application Settings, in the DHCP Server tab.

Once the DHCP process has been completed, the network settings can be set using the Ethernet Port Configuration via the Target Browser.

The Target Browser can be accessed under the Tools menu.

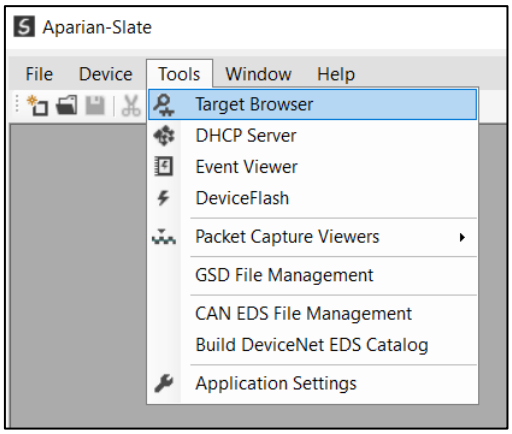


Figure 3.7. - Selecting the Target Browser

The Target Browser automatically scans the Ethernet network for EtherNet/IP devices.

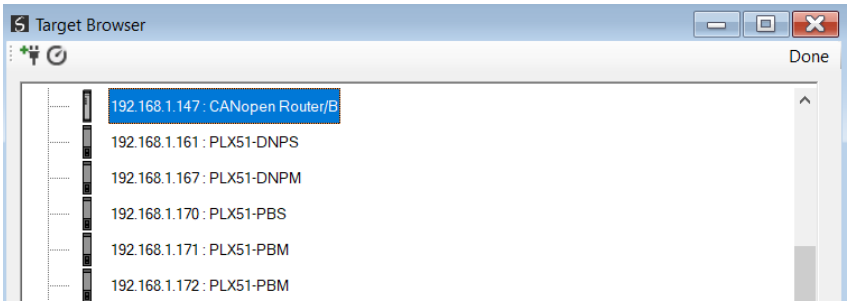


Figure 3.8. - Target Browser

Right-clicking on a device, reveals the context menu, including the Port Configuration option.

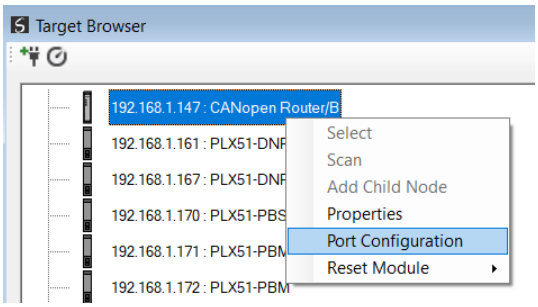


Figure 3.9. - Selecting Port Configuration

All the relevant Ethernet port configuration parameters can be modified using the Port Configuration window.

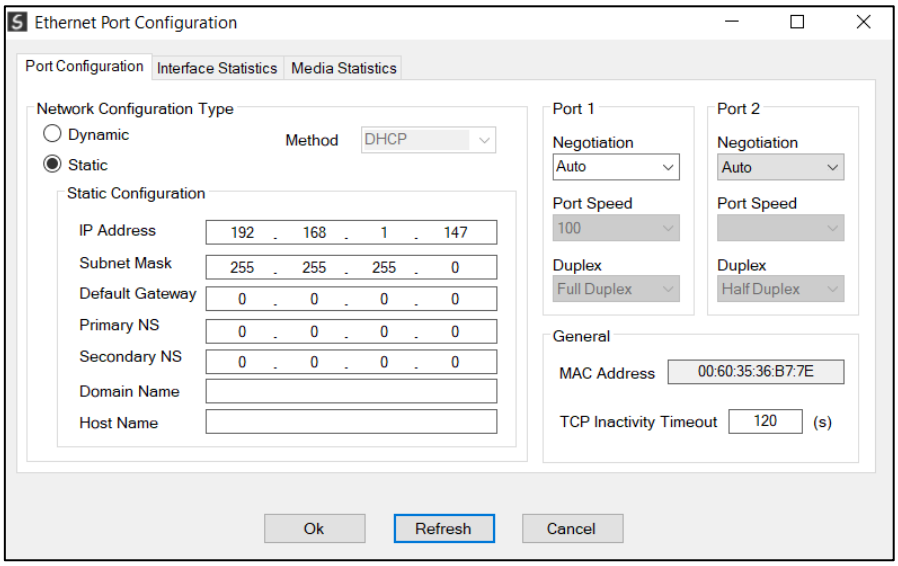


Figure 3.10. - Port Configuration

Alternatively, these parameters can be modified using Rockwell Automation's RSLinx software.

3.3. CREATING A NEW PROJECT

Before the user can configure the module, a new Slate project must be created. Under the File menu, select New.

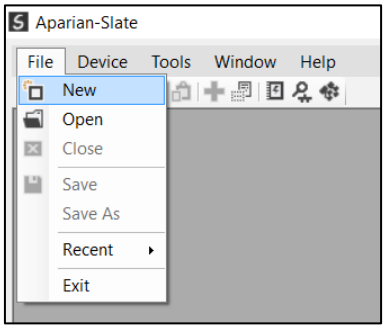


Figure 3.11. - Creating a new project

A Slate project will be created, showing the Project Explorer tree view. To save the project use the Save option under the File menu.

A new device can now be added by selecting Add under the Device menu.

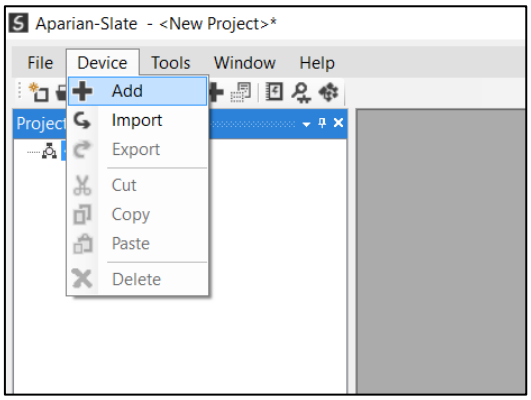


Figure 3.12. - Adding a new device

In the Add New Device window select the CANopen Router and click the Ok button.

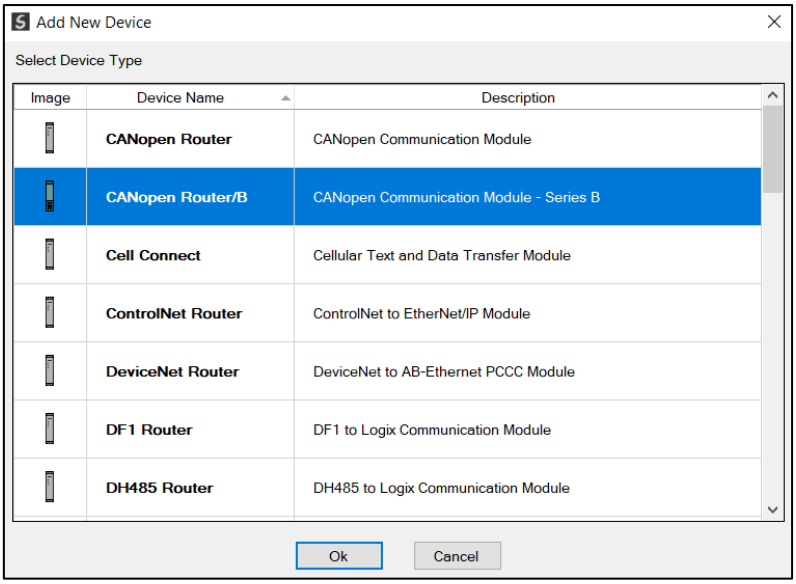


Figure 3.13 – Selecting a new CANopen Router

The device will appear in the Project Explorer tree as shown below, and its configuration window opened. The device configuration window can be reopened by either double clicking the module in the Project Explorer tree or right-clicking the module and selecting *Configuration*.

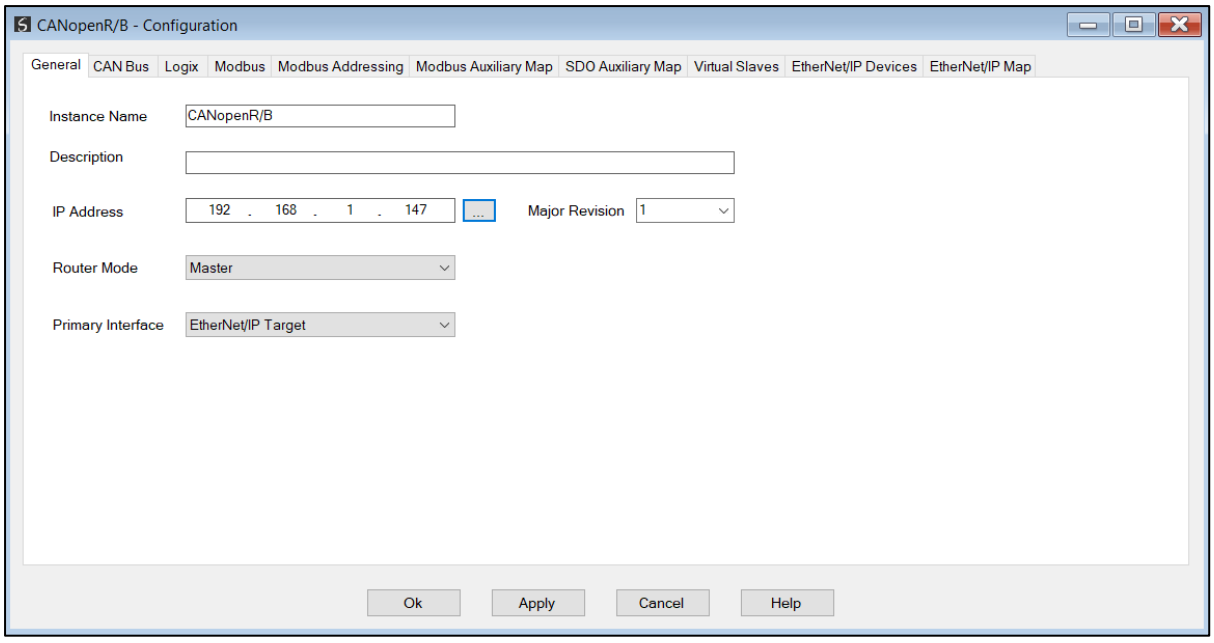


Figure 3.14. – CANopen Router configuration

Refer to the additional information section in this document for Slate's installation and operation documentation.

3.4. CANOPEN ROUTER PARAMETERS

The CANopen Router/B parameters are configured using Slate. **Refer** to the additional information section for documentation and installation links for Aparian Slate. The CANopen Router/B parameter configuration consists of a general configuration, CAN Bus configuration, Logix, Modbus, SDO mapping, Explicit EIP device, and Virtual Device Map (for operating as a CANopen slave). When downloading this configuration into the module it will be saved in non-volatile memory that persists when the module is powered down.

NOTE: When a firmware upgrade is performed, the module will clear all CANopen Router/B configuration and routing maps.

3.4.1. GENERAL

The general configuration is shown in the figure below. The CANopen Router/B general configuration window is opened by either double clicking on the module in the tree or right clicking the module and selecting *Configuration*.

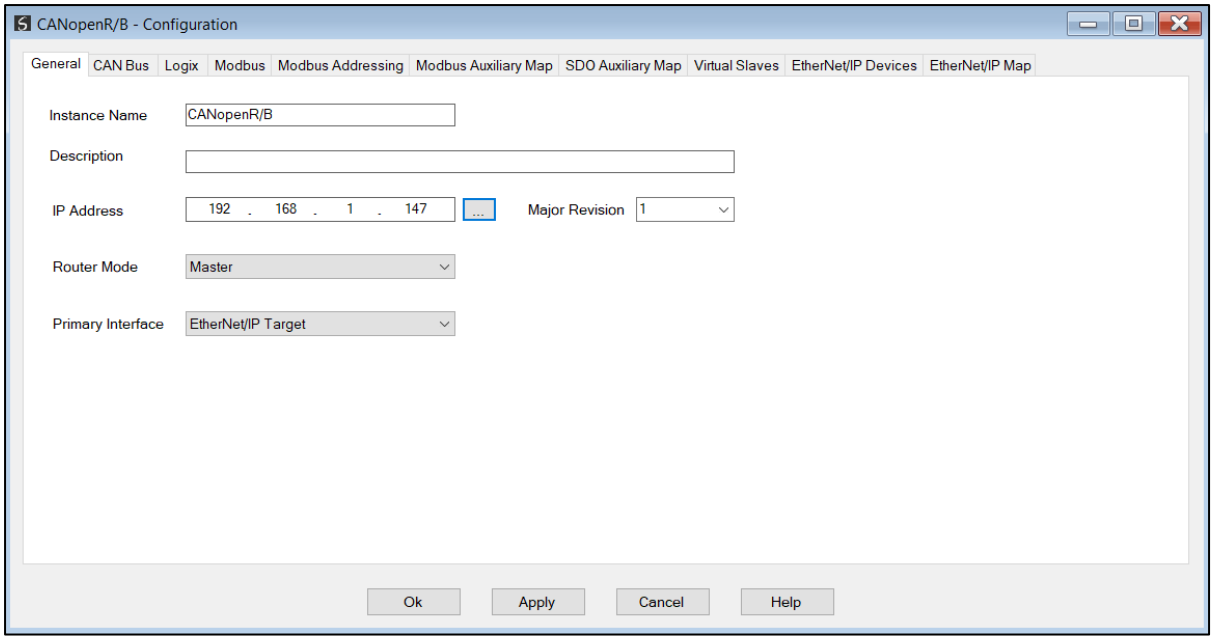


Figure 3.15. - General Configuration

The general configuration consists of the following parameters:

Parameter	Description
Instance Name	This parameter is a user defined name to identify the CANopen Router/B module instance.

Description	This parameter is used to provide a more detail description of the application for the module.
IP Address	The IP address of the target module. The user can use the target browse button to launch the target browser to the select the CANopen Router/B on the network.
Major Revision	The major revision of the module
Router Mode	**Master** The CANopen Router/B will operate as a CANopen Master on the CANopen network. **Slave** The CANopen Router/B will operate as a CANopen Slave on the CANopen network. The module can emulate multiple node addresses on a single CANopen network.
Primary Interface	Specifies the module's primary communication interface: **EtherNet/IP Target** The CANopen Router will be an EtherNet/IP target and exchange data with a Logix controller using Class 1 cyclic and/or Class 3 explicit communication. The Class 3 explicit messages will be via direct-to-tag data exchange with the Logix controller. The data from the Logix controller(s) will be available for CANopen Rx and Tx data. **Modbus Slave** The CANopen Router will be a Modbus Slave and can exchange data with a remote Modbus Master. The module can communicate on Ethernet TCP, RTU232, and RTU485. The Modbus data will be available for CANopen Rx and Tx data. **Modbus Master** The CANopen Router will be a Modbus Master and can exchange data with up to 20 remote Modbus Slaves and up to 100 mapping items. Each mapping item can communicate on either Modbus TCP, RTU232, or RTU485. The Modbus data will be available for CANopen Rx and Tx data. **EtherNet/IP Originator** The CANopen Router will be a EtherNet/IP Class 1 cyclic and/or Class 3 explicit connection originator. The module can have up to 5 Class 1 cyclic connections to Ethernet IP devices. The module can also read and write standard or custom CIP data to up to 5 EtherNet/IP devices using Class 3 or unconnected (UCMM) explicit connections while allowing up to 50 mapped items.

Table 3.1 - General configuration parameters

3.4.2. CAN Bus

The CAN Bus configuration is shown in the figure below. The CAN Bus configuration window is opened by either double clicking on the module in the tree or right-clicking the module and selecting *Configuration*.

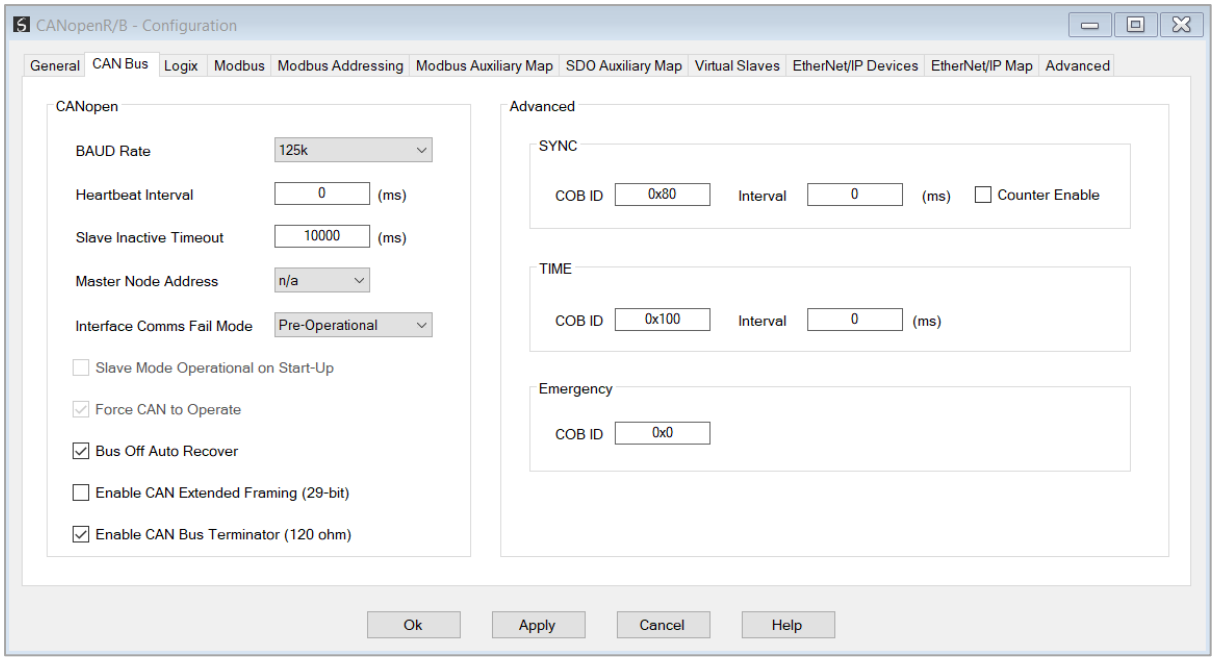


Figure 3.16 – CAN Bus Configuration

The CANopen Communication configuration consists of the following parameters:

Parameter	Description
BAUD Rate	The CANopen bus BAUD rate. The following options are available: • 10k • 20k • 50k • 125k • 250k • 500k • 800k • 1M
Heartbeat Interval	This is the rate (in milliseconds) at which the CANopen Router will send out heartbeat messages on the CANopen bus. This heartbeat will be sent for each node configured in the Virtual Slave Mapping at the configured interval. To disable sending of CANopen heartbeat messages the user can set this value to zero. **(CANopen Slave Mode Only)**
Slave Inactive Timeout	**When operating as a CANopen Master.** The amount of time (in milliseconds) elapsed since the last communication from a specific CANopen slave before the CANopen Router will indicate that the device is offline.

	When operating as a CANopen Slave. The amount of time (in milliseconds) since the last communication from the CANopen Master before the CANopen Router module will indicate that the CANopen Master is offline.
Master Node Address	The CANopen node address used by the CANopen Router/B (when operating as a CANopen Master), to send Heartbeat messages. **(CANopen Master Mode Only)**
Inhibit On Comms Fail Mode	The parameter will force the CANopen Router to a specific CANopen communication mode when the primary interface (EtherNet/IP Target, Modbus Slave, Modbus Master or EtherNet/IP Originator) communication has failed. **Pre-Operational** The CANopen network will be put into pre-operational mode when in CANopen Master Mode or the module will go into pre-operational mode when in CANopen Slave Mode. Pre-operational mode does not allow any process data (PDO) to be exchanged. **Inhibit** All communication on the CANopen network will be stopped. **NOTE: This is only valid when the Ethernet communication is not set to Modbus Master or EtherNet/IP Originator.** **Normal** There will be no change in the CANopen communication and the module will keep the CANopen network in the last state it was set. **NOTE:** The primary interface communication will indicate a fail when any of the communication devices or any of the mapping items have note successfully completed a data exchange.
Slave Mode Operational on Start-Up	When this option is set the module (in slave mode) will start up in operational mode rather than the normal pre-operational. **(CANopen Slave Mode Only)**
Bus Off Auto Recover	Enable or Disable automatic CANopen network recovery when the module has detected that the CAN bus network is off.
Enable CAN Extended Framing (29-bit)	When this option is selected, the CANopen Router/B module will transmit and receive 29-bit extended CAN messages when communicating on the CANopen network.
Enable CAN Bus Terminator	Enables or disables the 120 Ohm terminator on the CAN bus network. The CAN bus network must be terminated at the two extremities of the network (i.e., the start and end of the network).
SYNC	**COB ID** The base address to use when sending and receiving SYNC messages on the CANopen network. **Interval (CANopen Master Mode Only)**

	This is the rate (in milliseconds) at which the CANopen Router will send out SYNC messages on the CANopen bus. To disable the sending of CANopen SYNC messages the user can set this value to zero. **Counter Enable (CANopen Master Mode Only)** When Counter Enable is set, then a counter value will be added to the SYNC message which is incremented each time a SYNC message is sent. This will enable the optional parameter used to define an explicit relationship between the current SYNC cycle and PDO transmission.
TIME	**COB ID** The base address to use when sending and receiving TIME messages on the CANopen network. **Interval (CANopen Master Mode Only)** This is the rate (in milliseconds) at which the CANopen Router will send out TIME messages on the CANopen bus. To disable the sending of CANopen TIME messages the user can set this value to zero.
Emergency	**COB ID** The base address to use when sending and receiving EMCY messages on the CANopen network.

Table 3.2 – CANopen Communication parameters

3.4.3. LOGIX

The Logix configuration is shown in the figure below. The Logix configuration is used when the module is operating as a EtherNet/IP target. The Logix controller selected will be used when performing direct-to-tag reads or writes for either SDOs or PDOs. The Logix configuration window is opened by either double clicking on the module in the tree or right-clicking the module and selecting *Configuration*.

NOTE: See the Logix Target section for more detail regarding the Logix Target configuration and operation.

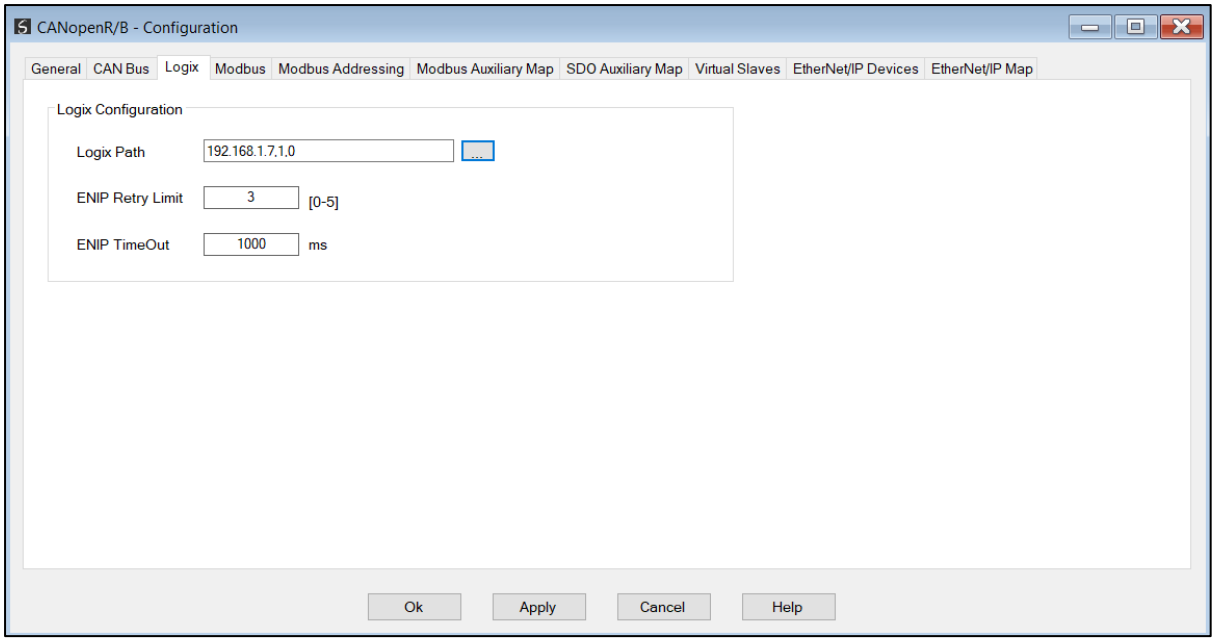


Figure 3.17 Logix Configuration

The Logix configuration (used for Class 3 Direct-To-Tag communication) consists of the following parameters:

Parameter	Description
Logix Path	The CIP path to the Logix controller which will be used to exchange data with the various CANopen devices. The user can use the target browse button to launch the target browser to select the Logix controller on the network.
ENIP Retry Limit	The number of EtherNet/IP retries the module will make once no response has been received from the Logix Controller.
ENIP Timeout	The time in milliseconds after which a retry is sent. Once the first retry is sent the next retry will be sent after the same amount of time. This will repeat until the ENIP Retry Limit is reached.

Table 3.3 – Logix parameters

3.4.4. MODBUS

The Modbus configuration is shown in the figure below. The Modbus configuration is relevant when the module has a Modbus Master or Modbus Slave operating interface. The Modbus configuration window is opened by either double clicking on the module in the tree or right-clicking the module and selecting *Configuration*.

NOTE: See the Modbus section for more detail regarding the Modbus configuration and operation.

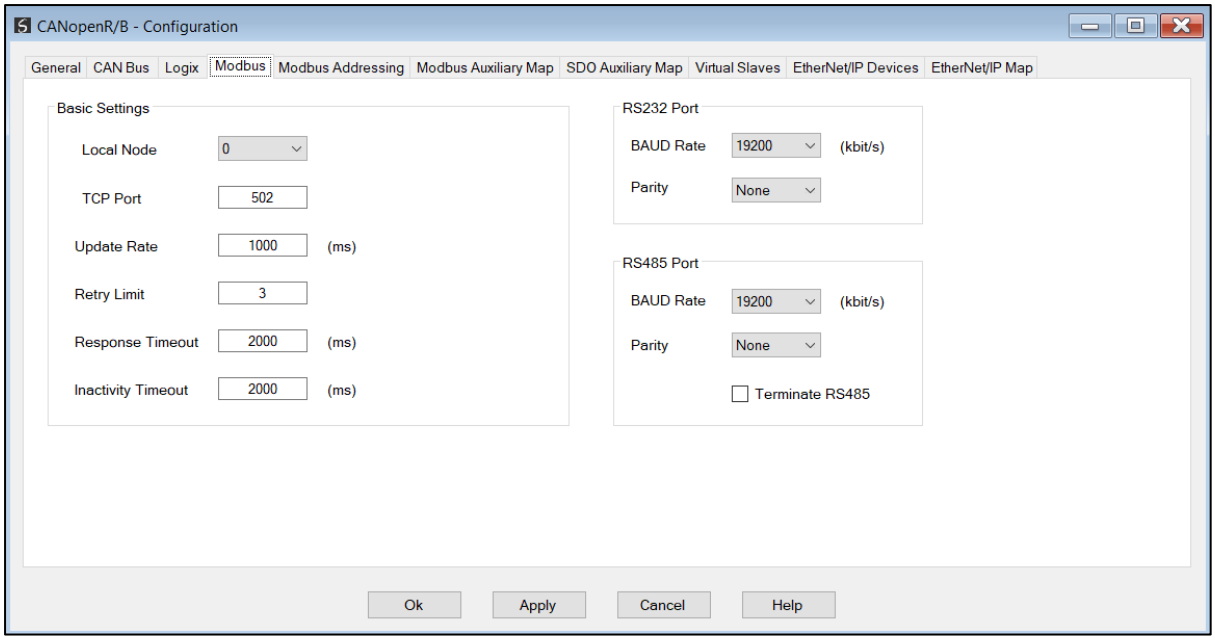


Figure 3.18 – Modbus Configuration

The Modbus Communication configuration consists of the following parameters:

Parameter	Description
Local Node	The Modbus Node address assigned to the CANopen Router.
Modbus TCP Port	The TCP port to be used for the Modbus communication. By default the module will use the standard TCP port 502.
Update Rate	The period (in milliseconds) between master requests to the target Modbus device. **(Modbus Master only)**
Retry Limit	The number of successive Modbus request retries before the request is set to have failed. **(Modbus Master only)**
Response Timeout	The time (in milliseconds) the module will wait for a valid Modbus response. **(Modbus Master only)**
Inactivity Timeout	The amount of time during which no Modbus requests have been received before the CANopen Router indicates that the connection to the Modbus Master is no longer active. **(Modbus Slave only)**
RS232 Port	
BAUD Rate	The RS232 serial port's BAUD rate . (Modbus RTU232)
Parity	The RS232 serial port's Parity configuration. (Modbus RTU232)
RS485 Port	

BAUD Rate	The RS485 serial port's BAUD rate . (Modbus RTU485)
Parity	The RS485 serial port's Parity configuration. (Modbus RTU485)
Terminate RS485	Enables the on-board 125Ω RS485 terminating resistor.

Table 3.4 – Modbus parameters

3.4.5. MODBUS ADDRESSING

The Modbus Addressing configuration is shown in the figure below. The Modbus Addressing configuration is relevant when the module has a Modbus Master or Modbus Slave operating interface. The Modbus Addressing configuration window is opened by either double clicking on the module in the tree or right-clicking the module and selecting *Configuration*.

NOTE: See the Modbus section for more detail regarding the Modbus configuration and operation.

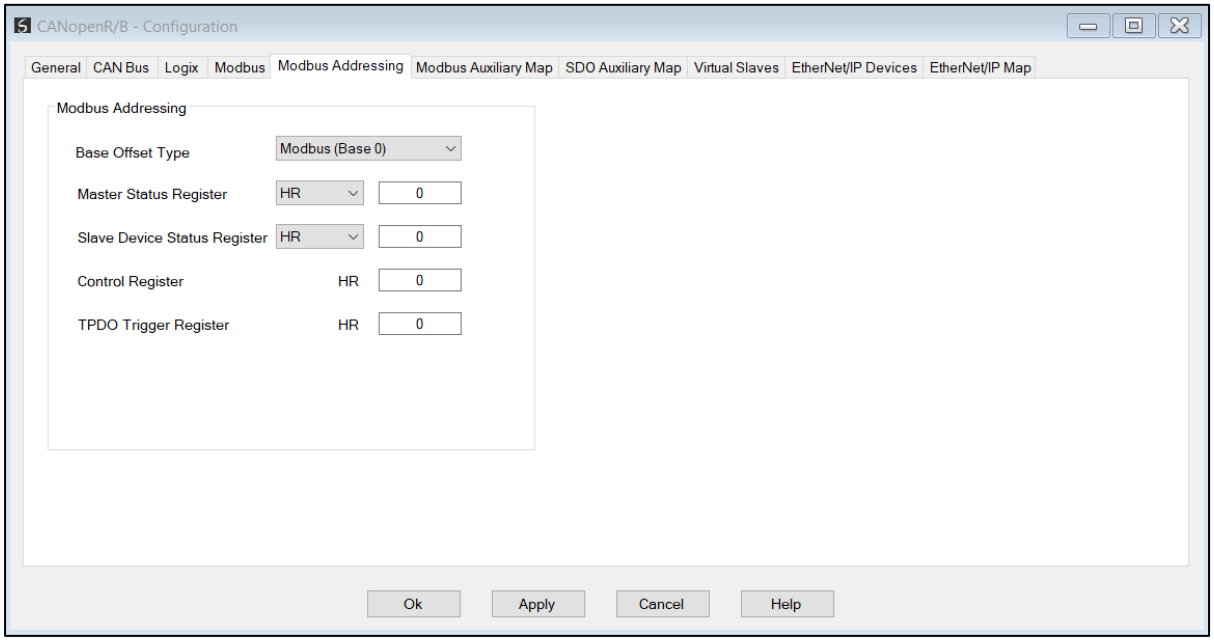


Figure 3.19 – Modbus Addressing Configuration

The Modbus Addressing configuration consists of the following parameters:

Parameter	Description
Base Offset Type	Base Address Offset Type Modbus (Base 0) – Conventional Modbus addressing where the first address is 0. PLC (Base 1) – PLC addressing, where the first address is 1.

Master Status Register	The Modbus Holding or Input Register address starting offset where the Master Status will be updated. See the Modbus Operation in section 6.3.2 or section 6.4.2.
Slave Device Status Register	The Modbus Holding or Input Register address starting offset where the Slave Device Status will be updated. See the Modbus Operation in section 6.3.2 or section 6.4.2.
Control Register	The Modbus Holding address starting offset where the Module Control will updated. See the Modbus Operation in section 6.3.2 or section 6.4.2.
TPDO Trigger Register	The Modbus Holding address starting offset where the TPDO Trigger Control will updated. See the Modbus Operation in section 6.3.2 or section 6.4.2.

Table 3.5 – Modbus Addressing parameters

3.4.6. MODBUS AUXILIARY MAP

The Modbus Auxiliary Map configuration is shown in the figure below. The Modbus configuration is only applicable when the module has a Modbus Master operating interface. Up to 100 mapping items can be configured while communicating to up to 20 Modbus Slave devices. The Modbus Auxiliary Map configuration window is opened by either double clicking on the module in the tree or right-clicking the module and selecting *Configuration*.

NOTE: See the Modbus section for more detail regarding the Modbus configuration and operation.

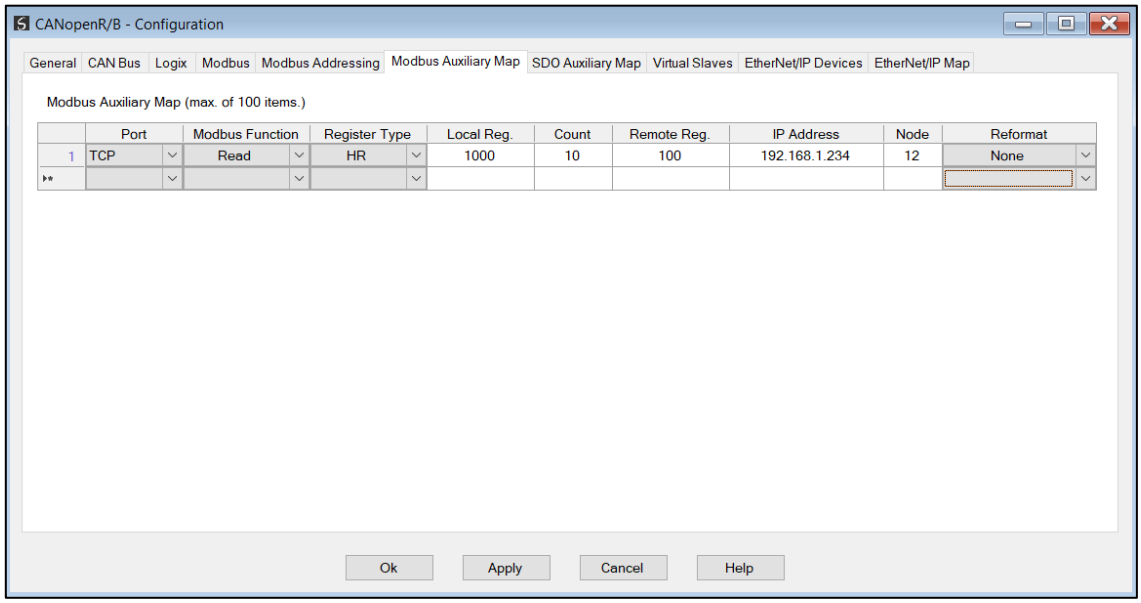


Figure 3.20 – Modbus Auxiliary Map Configuration

The Modbus Auxiliary Map configuration consists of the following parameters:

Parameter	Description
Port	The external port to be used: **TCP** – Modbus TCP (Ethernet) **RS232** – Modbus RTU232 **RS485** – Modbus RTU485
Modbus Function	This is the Modbus function that is sent to the Modbus Slave. **Read** – Read a Modbus Register (e.g. HR, IR, CS, or IS) from a Modbus Slave. **Write** – Write a Modbus Register (e.g. HR or CS) to a Modbus Slave.
Register Type	Modbus Register Type: **CS** – Coil Status **IS** – Input Status **IR** – Input Register **HR** – Holding Register
Local Reg.	The local (internal) CANopen Router/B Modbus register address.
Count	The number of Modbus elements to read or write.
Remote Reg.	The remote slave Modbus address register.
IP Address	The IP address of the remote Modbus TCP slave.
Node	The Modbus Node address of the remote Modbus slave.
Reformat	Used to specify how the data is formatted before writing to, or after reading from, the Modbus slave. **None** – No reformatting applied. (AA BB CC DD). **BB AA** – 16bit Byte swap **BB AA DD CC** – 32bit Byte Pair Swap **CC DD AA BB** – Word Swap **DD CC BB AA** – Word and Byte Pair Swap

Table 3.6 – Modbus Auxiliary Map parameters

3.4.7. SDO Auxiliary Map

The Service Data Object (SDO) Auxiliary Map configuration is shown in the figure below. The SDO mapping can be used when the module is operating as a CANopen Master. SDOs can be read from, or written to, CANopen Slave devices asynchronous to the PDO data. The SDO Auxiliary Map configuration window is opened by either double clicking on the module in the tree or right-clicking the module and selecting *Configuration*.

NOTE: See the SDO mapping section for more detail regarding the SDO exchange configuration and operation.

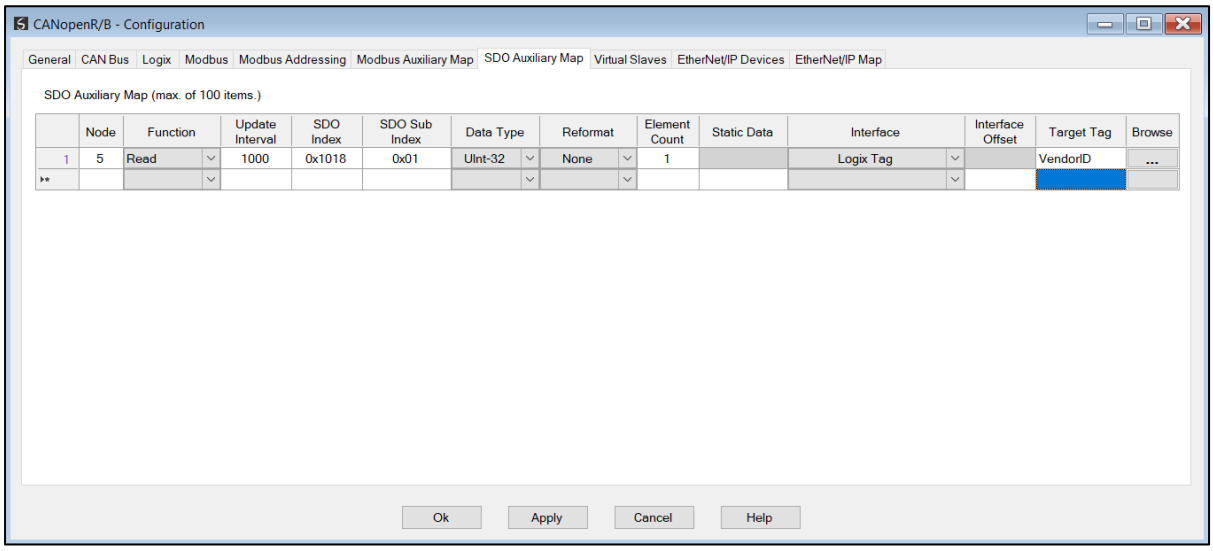


Figure 3.21 – SDO Auxiliary Map Configuration

The SDO Auxiliary Map configuration consists of the following parameters:

Parameter	Description
Node	The destination CANopen Slave node address.
Function	The CANopen function that is sent to the CANopen Slave. **Read** – Read a SDO parameter (index and sub-index) from a CANopen Slave. **Write** – Write a SDO parameter (index and sub-index) to a CANopen Slave. **Write Static** – Write a static value to a CANopen Slave SDO (index and sub-index). The value to be written will be in the *Static Data* parameter field in the table (see below).
Update Interval	The interval (in milliseconds) at which the mapped item will be executed.
SDO Index	The index number of the SDO. Note that this number is normally displayed in hexadecimal format (e.g. 0x101A), but can be entered in decimal or hexadecimal notation (0x..).
SDO Sub Index	The sub-index number of the SDO. Note that this number is normally displayed in hexadecimal format (e.g. 0x0A), but can be entered in decimal or hexadecimal notation (0x..).
Data Type	The Data Type of the SDO to be accessed.
Reformat	How the data is formatted before reading or writing from/to the CANopen slave. **None** – No reformatting will be done. **BB AA** – 16bit Byte swap **BB AA DD CC** – 32bit Byte Pair Swap **CC DD AA BB** – Word Swap

	DD CC BB AA – Word and Byte Pair Swap
Element Count	The number of elements for the specific SDO. For example if there are two Float numbers to be read, then the data type will be *Real-32* and the element count will be 2. If there are four 16bit integers to be read, then the data type will be *int16* and the element count will be 2.
Static Data	When the Write Static Data function has been selected, the value in the field will be written to the CANopen Slave SDO parameter.
Interface	The interface selection will be used to provide the source or destination of the CANopen SDO data based on the Primary Interface selected: **EtherNet/IP Target** The user can select either a Logix tag or the input/output assemblies from the Logix Class 1 connection (***Class 1 Conn.3*** or ***Class 1 Conn.4***) **Modbus Slave/Master** Modbus Register (**HR**, **IR**, **IS** or **CS**) and the **Offset** in the Modbus Register. **EtherNet/IP Originator** The user can select either the internal **Explicit Map** (where the Explicit EtherNet/IP data exchange resides) or one of the Class 1 EtherNet/IP originator connections that have been added to the CANopen Router module.

Table 3.7 – SDO Auxiliary Map parameters

3.4.8. VIRTUAL SLAVES

The Virtual Slaves Map configuration is shown in the figure below. The Virtual Slaves are used when the module is operating as a CANopen Slave. The mapping is used to emulate PDOs from configurable CANopen Slaves. Up to 125 mapped items can be used to emulate up to 125 CANopen devices. The Virtual Slaves configuration window is opened by either double clicking on the module in the tree or right-clicking the module and selecting *Configuration*.

NOTE: See the CANopen Slave Mode section for more detail regarding the Virtual Slaves Map configuration and operation.

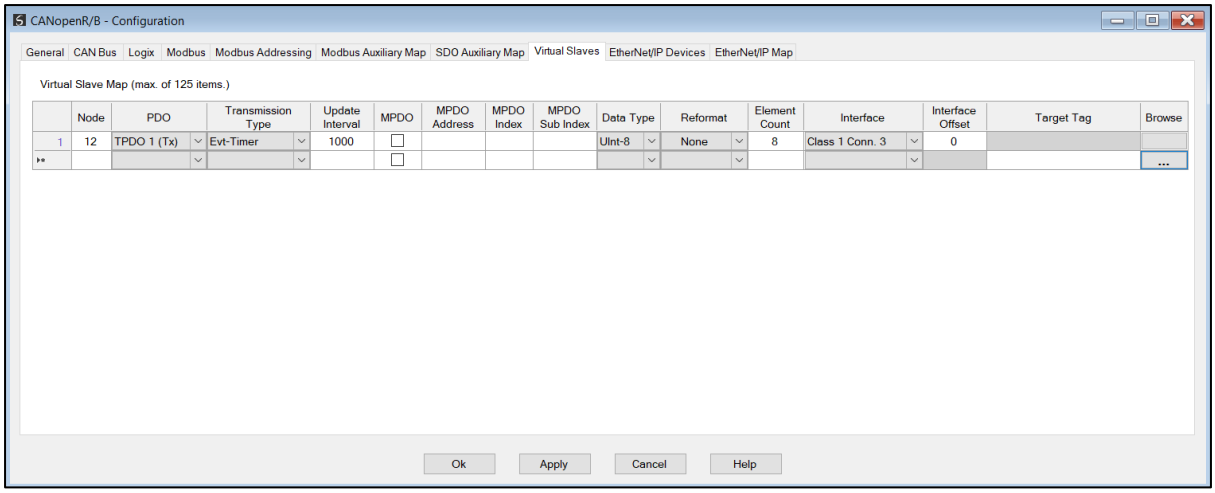


Figure 3.22 – Virtual Slaves Map Configuration

The Virtual Slave Map configuration consists of the following parameters:

Parameter	Description
Node	The CANopen Slave node address the CANopen Router will emulate. The module can emulate up to 125 slaves.
PDO	There are two functions supported for mapping PDOs (process variables) for the CANopen Router when operating as a CANopen slave. **TPDO x (Tx)** TPDOs are the PDOs **sent to** the CANopen Master. **RPDO x (Rx)** RPDOs are the PDOs **received from** the CANopen Master. Where x can be between 1 and 4.
Transmission Type	Specify the Transmission Type as one of the following: **Sync** The CANopen Router in Slave mode will send out the PDO data to the CANopen Master once a ***SYNC*** packet has been received. **RemoteTxReq (TPDO only)** The CANopen Router will transmit the PDO to the CANopen Master when requested from the CANopen Master. **Evt-Timer (TPDO only)** The CANopen Router in Slave mode will send out the PDO data to the CANopen Master every ***Update Interval***. **Evt-Interface (TPDO only)** The CANopen Router in Slave mode will send out the PDO data to the CANopen Master every time the relevant PDO bit is toggled in the Slave Triggers for the respective Primary interfaces. **Evt-Manufacturer (RPDO only)**

	This is set if the CANopen Master sends PDO data based on a specific event (e.g. Timer).
Update Interval	The update rate (in milliseconds) at which the PDOs will be sent (when transmission type is ***Evt-Timer*** or ***Remote Tx Request***).
MPDO	Multiplexing Each PDO can be multiplexed (if supported by the CANopen Master) to have multiple process variables associated with it. With normal PDOs each PDO has a maximum of 8 bytes while with multiplexed PDOs each multiplexed process variable has maximum of 4 bytes. To enable Multiplexing the user must select the MPDO checkbox in the mapping of the PDO. **MPDO Address** The address of the process variable in the PDO. **MPDO Index** The index of the process variable in the PDO. **MPDO Sub Index** The sub index of the process variable in the PDO.
Data Type	The data type to be used when copying to/from the Primary Interface.
Reformat	How the data is formatted before writing to, or after reading from, the CANopen slave. **None** – No reformatting will be done. **BB AA** – 16bit Byte Pair Swap **BB AA DD CC** – 32bit Byte Pair Swap **CC DD AA BB** – Word Swap **DD CC BB AA** – Word and Byte Pair Swap
Element Count	The number of elements to be used for the specific PDO. For example, the user can have 2 x 32-bit real values or 8 x 8-bit integers.  **NOTE:** The element count must be such that the total byte count (element count multiplied by the data type size) must not exceed: **8 bytes** for **non-multiplexed** mapped items, and **4 bytes** for **multiplexed** map items.
Interface	The interface selection will be used to provide the source or destination of the CANopen PDO data based on the Primary Interface selected: **EtherNet/IP Target** The user can select either a Logix tag or the input/output assemblies from the Logix Class 1 connection (***Class 1 Conn. 3*** or ***Class 1 Conn. 4***) **Modbus Slave/Master** Modbus Register (**HR**, **IR**, **IS**, or **CS**) and the ***Offset*** in the Modbus Register. **EtherNet/IP Originator**

	The user can select either the internal ***Explicit Map*** (where the Explicit EtherNet/IP data exchange resides) or one of the Class 1 EtherNet/IP originator connections that have been added to the CANopen Router module.

Table 3.8 – Virtual Slave Map parameters

3.4.9. EtherNet/IP Devices

This tab is enabled when the Primary Interface selected is EtherNet/IP Explicit Messaging.

The EtherNet/IP Devices configuration is shown in the figure below. Up to 5 EtherNet/IP devices can be configured with up to 50 EtherNet/IP mapped items allowing for either explicit EtherNet/IP Class 3 or Unconnected Messaging (UCMM) to any of the 5 configured devices. The data from each EtherNet/IP device is written to, or read from, an internal data table with a size of 100Kbytes. See the *Explicit EtherNet/IP Messaging* section for more details.

The EtherNet/IP Devices configuration window is opened by either double clicking on the module in the tree or right-clicking the module and selecting *Configuration*.

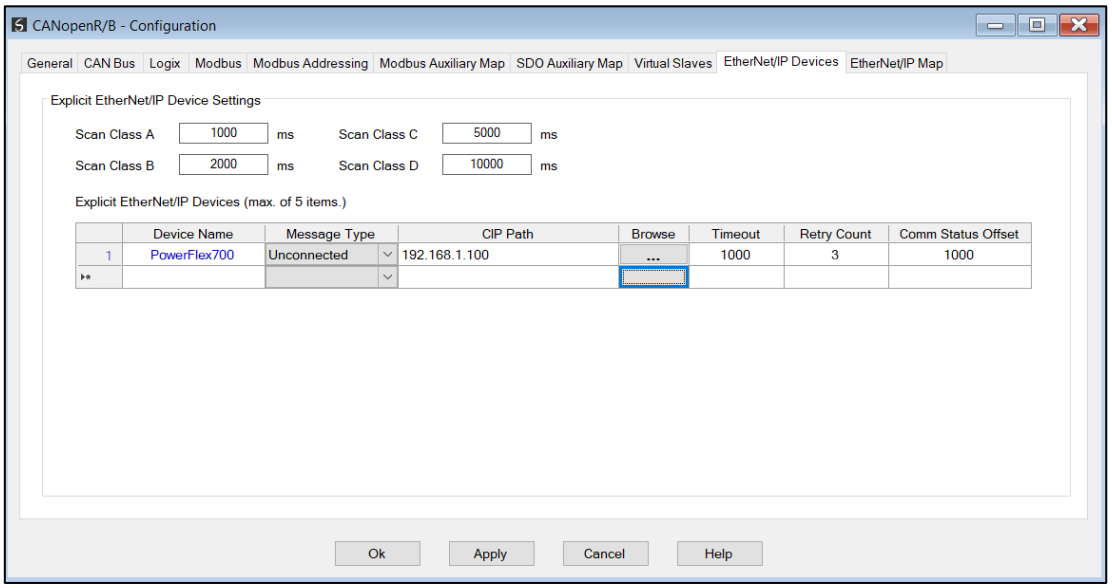


Figure 3.23 – EtherNet/IP Devices configuration

The EtherNet/IP Devices configuration consists of the following parameters:

Parameter	Description
Scan Class A, B, C, D	The configurable update rates (in milliseconds) for each mapped item in the EtherNet/IP Map.
Device List (per device)	

Device Name	The user assigned instance name for the specific device.
Message Type	The module can use either ***Class 3*** or ***Unconnected Messaging*** when communicating to the target EtherNet/IP device.
CIP Path	The CIP Path to the target device. This can either be entered manually or the user can browse to them by clicking the ***Browse*** button. The Target Browser will open and automatically scan for all available EtherNet/IP devices. If the Ethernet/IP module is a bridge module, it can be expanded by right-clicking on the module and selecting the ***Scan*** option. The required EtherNet/IP device can then be chosen by selecting it and clicking the ***Ok*** button, or by double-clicking on the target module.
Timeout	The amount of time (in milliseconds) the module will wait for a response from the target EtherNet/IP device.
Retry Count	The number of message retries before the target EtherNet/IP device is considered offline.
Comm Status Offset	This is the offset in the data table (used to map EtherNet/IP device data) which provides the communication status of each EtherNet/IP device. The Communication Status is as shown below: Bit 0 - (1:Device Online , 0:Device Offline) Bit 1 to 7 – Reserved.

Table 3.9 – EtherNet/IP Devices configuration parameters

3.4.10. ETHERNET/IP MAP

This tab is enabled when the Primary Interface selected is EtherNet/IP Explicit Messaging.

The EtherNet/IP Map configuration is shown in the figure below. Up to 50 EtherNet/IP mapped items, either explicit EtherNet/IP Class 3 or Unconnected Messaging (UCMM) to any of the 5 pre-configured devices can be configured. The data from each EtherNet/IP device is written to or read from a data table with a size of 100Kbytes. See the *Explicit EtherNet/IP Messaging Operation* section for more details.

The EtherNet/IP Map configuration window is opened by either double clicking on the module in the tree or right-clicking the module and selecting *Configuration*.

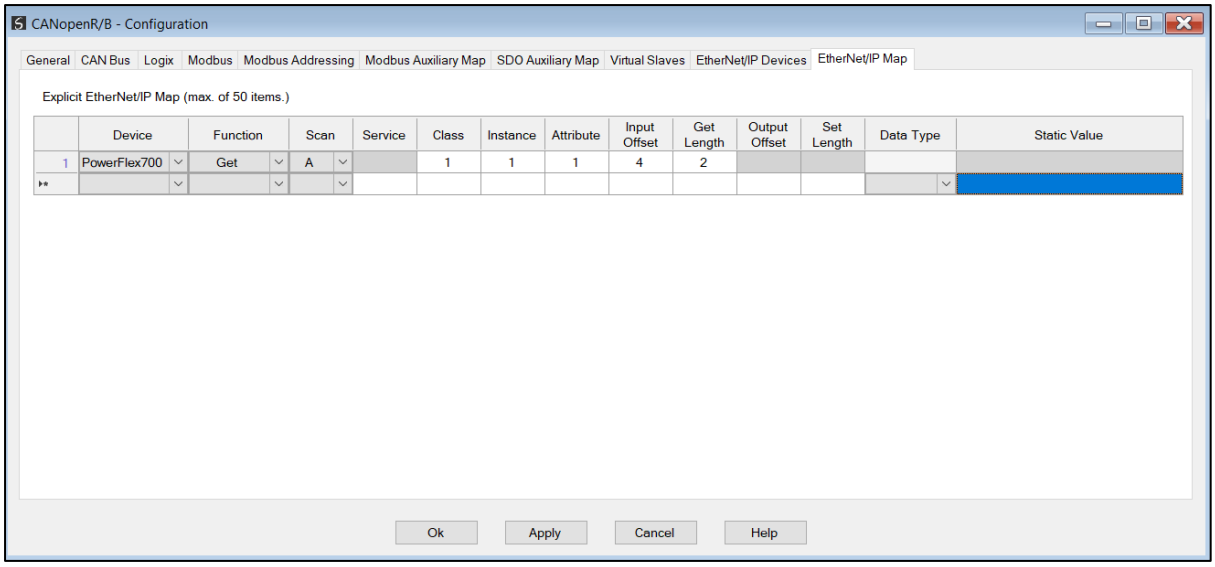


Figure 3.24 – EtherNet/IP Map configuration

The EtherNet/IP Map configuration consists of the following parameters:

Parameter	Description
Device	The device instance name configured in the previous EtherNet/IP Devices tab. The selected device will be used for executing the communication function.
Function	The user can select one of four functions. **Get** The module will read data from the target EtherNet/IP device by using the Get Single Attribute CIP function. The received data will be placed into the Data Table at the ***Input Offset*** location configured in this tab. **Set** The module will write data to the target EtherNet/IP device by using the Set Single Attribute CIP function. The data to be written will be retrieved from the Data Table at the ***Output Offset*** location configured in this tab. **Set Static** Similar to the Set function above, but the data to be written will be fixed (equal to the *Static Value*) parameter in this configuration window. This function will typically be used with the single (**S**) Scan class which means the CANopen Router module can be setup to write the fixed value only once when the target device communication has been established. **Custom** This function allows the user to use a custom Service to write and read data in the same transaction. The user will need to see which custom services that target device supports in that device's user manual.

Scan	The user can select Scan Class **A**, **B**, **C** or **D** (which was configured in the EtherNet/IP Devices tab). The specific mapped item will then be executed at that configured scan class rate. The user can also select the **S** class which means that the mapped item will only execute once when communication to the target device is established. If the target device goes offline, then the mapped items with this class will be re-armed, and resent when communication is re-established.
Service	The custom CIP service/function which is only available when the ***Custom*** function has been selected.
Class, Instance, Attribute	The CIP class, instance, and attribute of the request message to be sent.
Input Offset	The location in the internal Data Table where the received data will be written. This will only be available for ***Get*** and ***Custom*** functions.
Get Length	The length of the data to be received. If the number of bytes received is more than the Get Length, then the data will **not** be written to the internal Data Table. This will only be available for ***Get*** and ***Custom*** functions.
Output Offset	The location in the internal Data Table from where the data to be written to the target device will be read. This will only be available for ***Set*** and ***Custom*** functions.
Set Length	The length of the data to be written. This will only be available for ***Set*** and ***Custom*** functions.
Data Type	The data type of the Static Value. This will only be available for ***Set Static*** function.
Static Value	The value to be written to the target device when the ***Set Static*** function has been selected. **Note**: When using the SINT Array data type, the values must be entered as space-delimited hex values. For example: 05 34 2E A1

Table 3.10 – EtherNet/IP Map configuration parameters

3.4.11. ADVANCED

The Advanced configuration window is opened by either double clicking on the module in the tree or right-clicking the module and selecting *Configuration*.

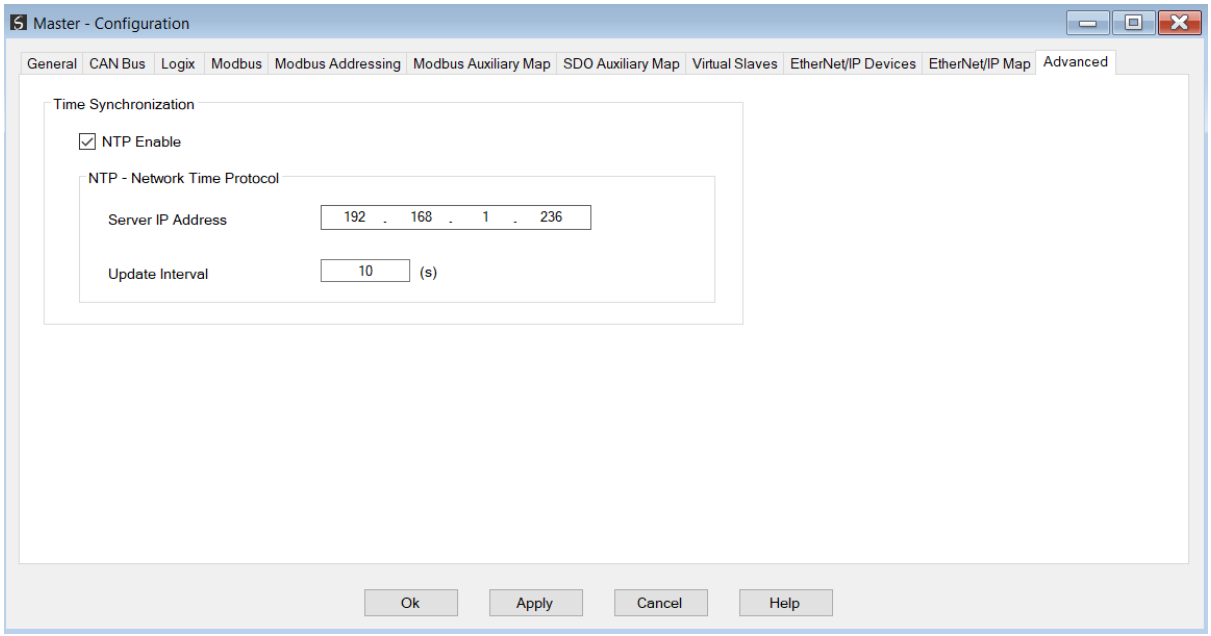


Figure 3.25 – Advanced configuration

The EtherNet/IP Map configuration consists of the following parameters:

Parameter	Description
NTP Enable	The CANopen Router/B can synchronize its onboard clock to an NTP Server by enabling NTP.
NTP – Server IP Address	This setting is the IP address of the NTP Server which will be used as a time source.
NTP – Update Interval	This setting is the updated interval (in seconds) that the CANopen Router/B will request time from the NTP Server.

Table 3.11 – Advanced configuration parameters

NOTE: When NTP is enabled, it will take precedence over all other time sources.
When the module is a CANopen Master with NTP enabled, then time information being received from the primary interface will be discarded.
When the module is a CANopen Slave with NTP enabled then time being received from the CANopen network will be discarded.

3.5. CANOPEN MASTER MODE

The module can be configured to operate as a CANopen Master on the CANopen network by selecting Master in the *Router Mode* parameter (see the *General Configuration*).

3.5.1. CAN EDS File Management

Each CANopen slave device has an EDS file that is required to provide information needed to configure the device for data exchange. Slate manages the CANopen EDS library which is used for adding devices to the CANopen Router when in Master mode.

The EDS File Management Tool is opened by selecting ***CAN EDS File Management*** under the ***Tools*** menu in the configuration utility.

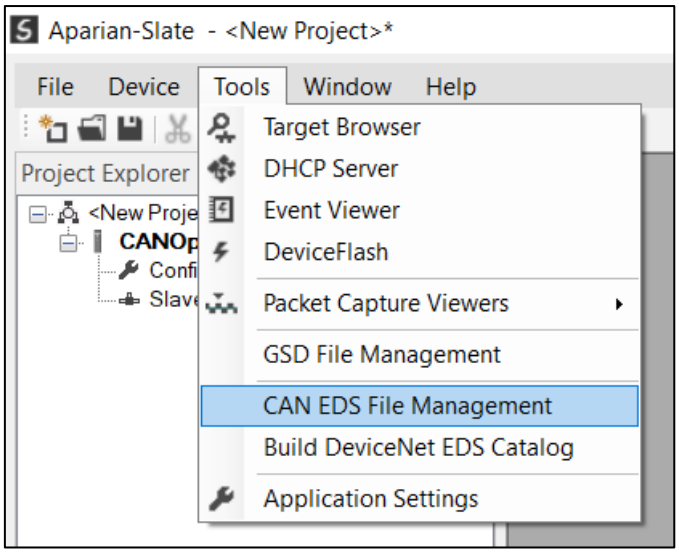


Figure 3.26 – Launching the CAN EDS File Management Tool

Once the tool has been opened a list of slave devices already registered using their CAN EDS files.

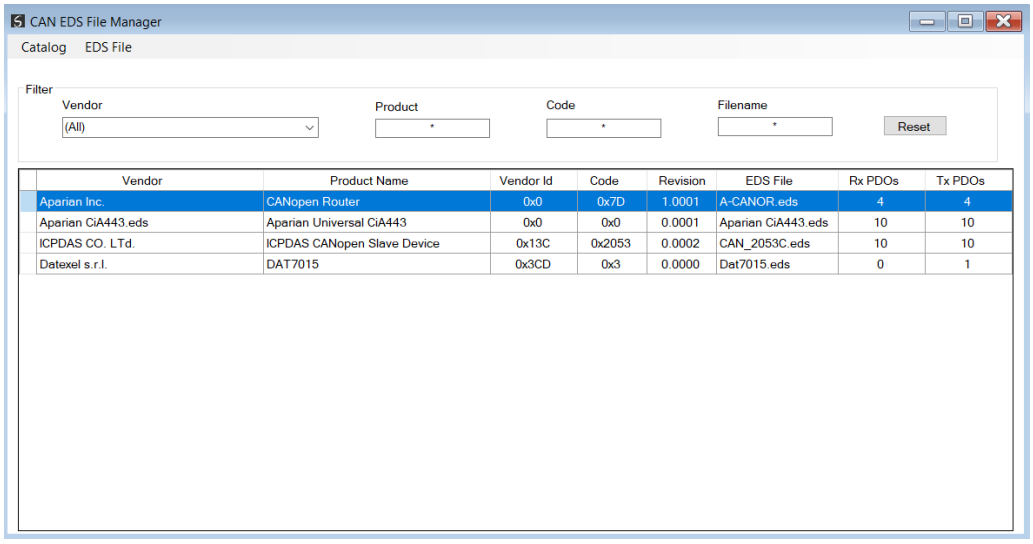


Figure 3.27 – CAN EDS File Management Tool

To add an EDS file the user will need to select the ***Add*** option under the EDS File menu.

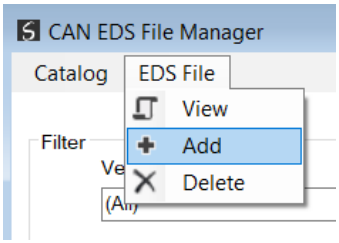


Figure 3.28 – CAN EDS File Adding

The required CAN EDS file will need to be selected as shown below:

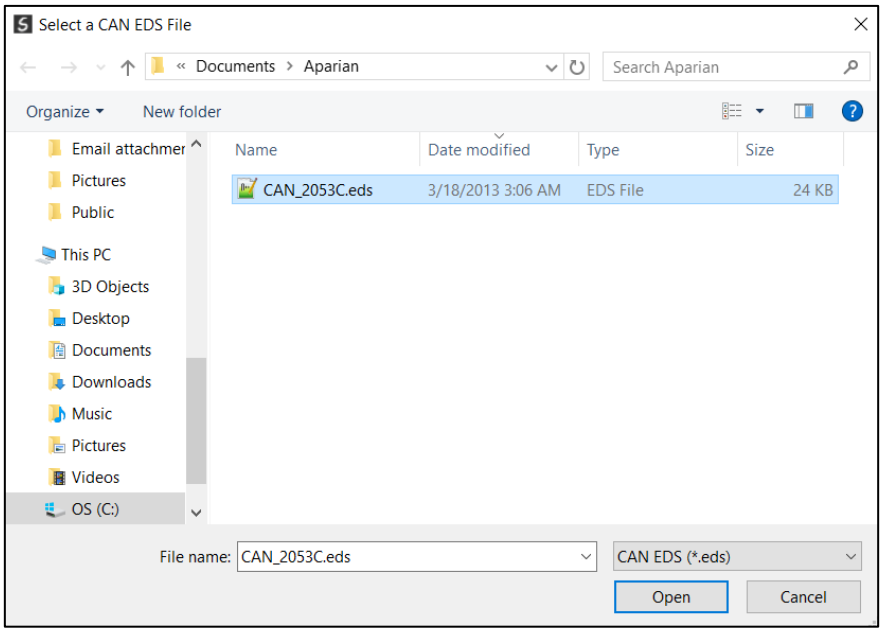


Figure 3.29 – CAN EDS File Adding

Once the file has been selected the CAN EDS File Management tool will add the slave device to the device list and recompile the CAN EDS catalog.

An entire CAN EDS catalog can be transferred from one system to another by using the Catalog ***Export*** and ***Import*** functions. This is done by selecting either ***Import*** or ***Export*** under the ***Catalog*** menu as shown below:

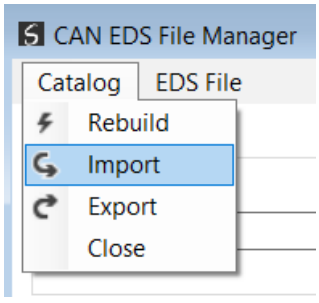


Figure 3.30 – CAN EDS Catalog importing

NOTE: Certain CANopen device manufacturers might not provide a CANopen EDS file. If this is the case the user can select the Generic CANopen EDS file provided in Slate to configure the module (see below):

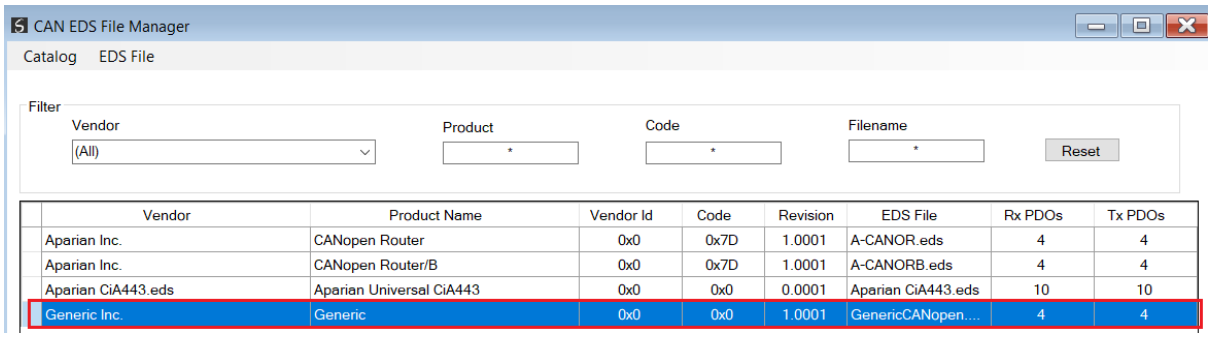


Figure 3.31 – CANopen Generic EDS file

3.5.2. Adding CANopen Slave Devices

The user will need to add each CANopen slave device to the CANopen Router which can then be configured. This can be done either manually or using the device discovery tool.

3.5.2.1. Manually

A CANopen device can be manually added by right-clicking on the ***Slave Devices*** item in the tree and selecting ***Add CANopen Device***.

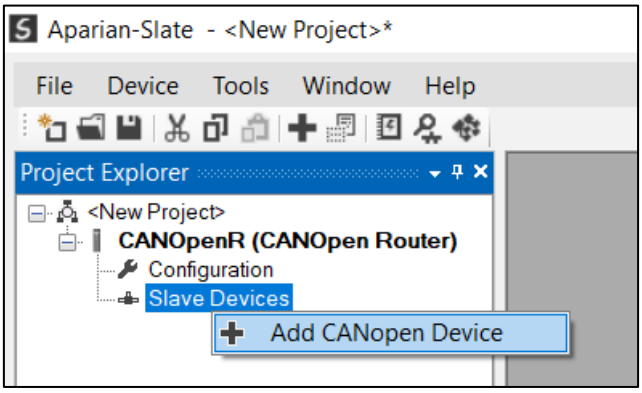


Figure 3.32 – Adding a CANopen Slave Device

The user will then need to select one of the CANopen devices from the CANopen EDS catalog in Slate.

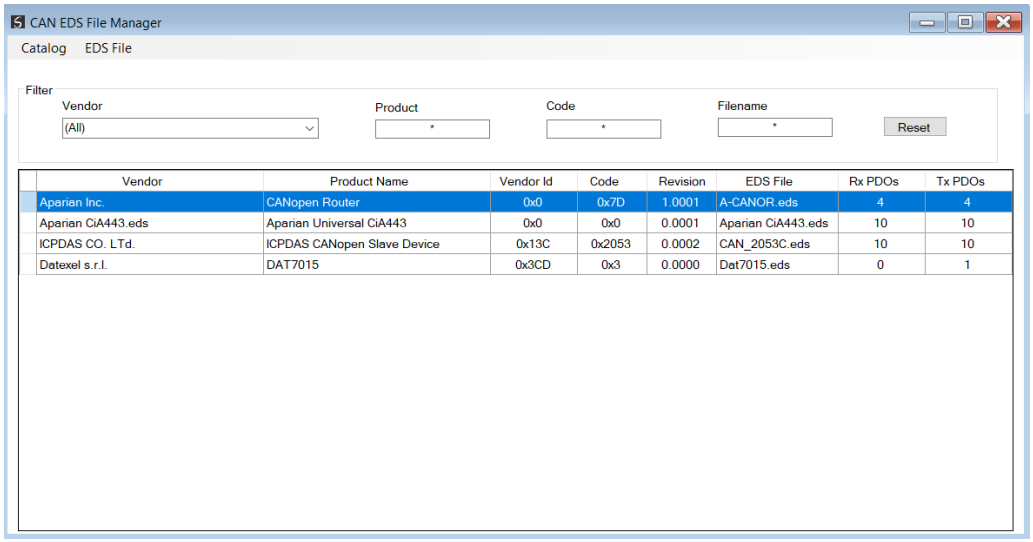


Figure 3.33 – Selecting a CANopen Slave Device from the EDS catalog

3.5.2.2. DEVICE DISCOVERY

The device discovery function scans the CANopen network and displays all the devices found on the network. This is done by opening the module status form and selecting the *Discovery* tab. Slate will start scanning the CANopen network for slave devices once the ***Start Discovery*** button has been pressed (see below).

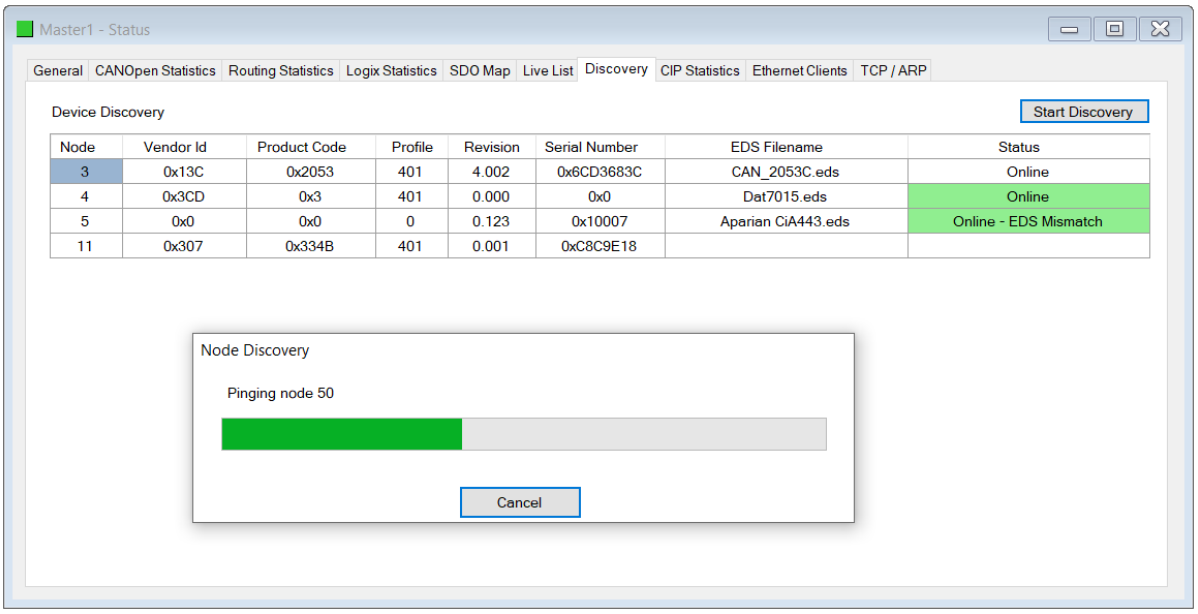


Figure 3.34 – Device Discovery

Once all the devices have been found the user will be able to add any of the devices to the CANopen Router Slave devices tree. This is done by right-clicking on the device in the discovery list and selecting *Add Device* (as shown below).

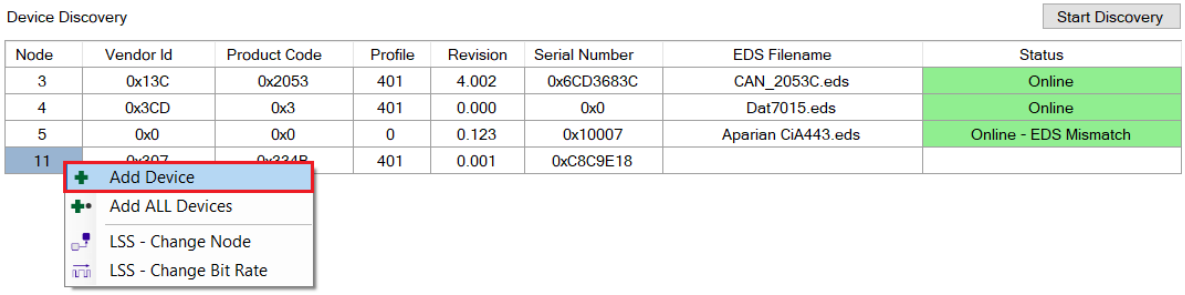


Figure 3.35 – Device Discovery – Add

NOTE: If a matching EDS file was not found during the device discovery, then the user will not be able to add the device from the discovery results. The user will then need to manually add a EDS file for the device.

3.5.3. LAYER SETTING SERVICES (LSS)

Certain devices do not support physical setting of their node address and/or BAUD rate, and require these parameters to be configured using the CANopen Layer Setting Services (LSS). The CANopen Router/B module supports LSS and can be accessed using the Device Discovery services.

3.5.3.1. CHANGE NODE ADDRESS

Once the device requiring the node address change was found in the Device Discovery window, the user can right-click on the device and select LSS - Change Node.

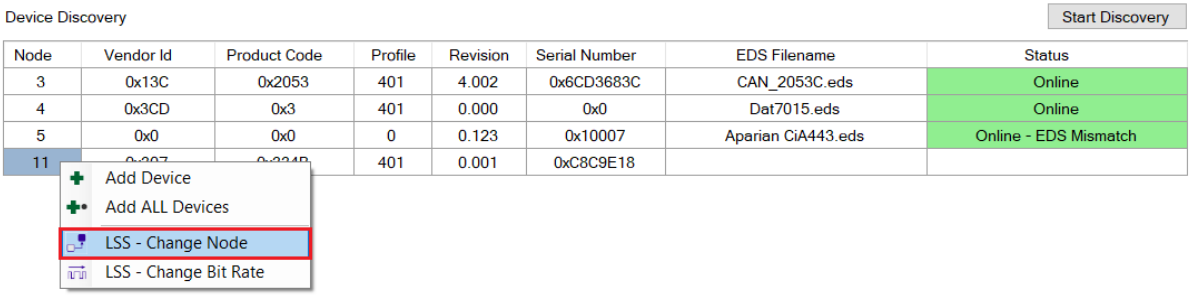


Figure 3.36 – LSS – Change Node Address

The user can then type in the new CANopen node number and once done rescan the CANopen network to find the CANopen Slave device at the new node address.

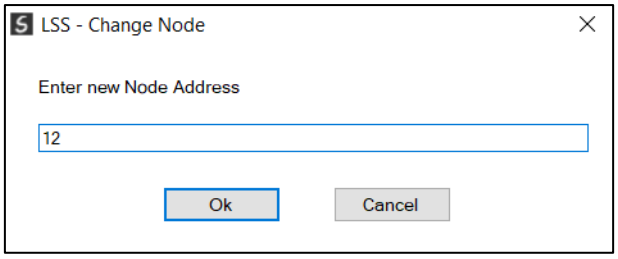
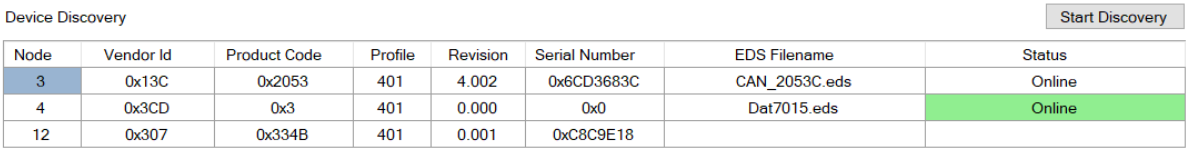


Figure 3.37 – LSS – Enter New Address

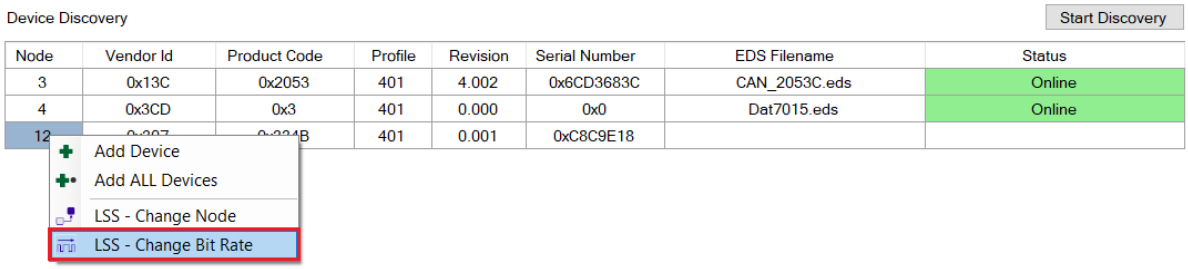


Node	Vendor Id	Product Code	Profile	Revision	Serial Number	EDS Filename	Status
3	0x13C	0x2053	401	4.002	0x6CD3683C	CAN_2053C.eds	Online
4	0x3CD	0x3	401	0.000	0x0	Dat7015.eds	Online
12	0x307	0x334B	401	0.001	0xC8C9E18		

Figure 3.38 – LSS – New Address Updated

3.5.3.2. CHANGE BIT RATE

Once the device requiring the BAUD rate change was found in the Device Discovery window, the user can right-click on the device and select *LSS - Change Bit Rate*.



Node	Vendor Id	Product Code	Profile	Revision	Serial Number	EDS Filename	Status
3	0x13C	0x2053	401	4.002	0x6CD3683C	CAN_2053C.eds	Online
4	0x3CD	0x3	401	0.000	0x0	Dat7015.eds	Online
12			401	0.001	0xC8C9E18		

Figure 3.39 – LSS – Change Bit Rate (BAUD)

The user can then select the new CANopen BAUD rate from the drop down list. Once done the user will need to **ensure that the CANopen Router/B BAUD rate has been updated to the new device BAUD Rate** and then rescan the CANopen network to find the CANopen Slave device operating at the new BAUD rate.

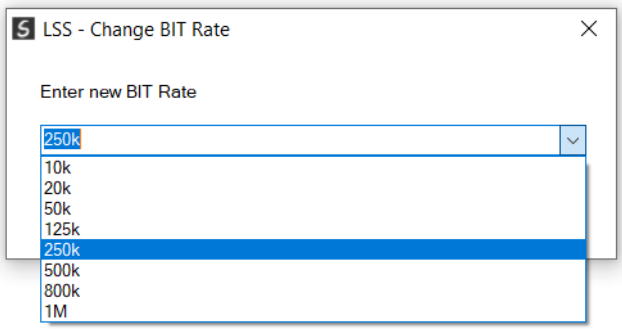


Figure 3.40 – LSS – Select new Bit Rate (BAUD)

3.5.4. CANopen Slave Device - General Configuration

The General configuration of the CANopen Slave Device is shown in the figure below. The Device General configuration window is opened by either double clicking on the slave device in the tree or right-clicking the slave device and selecting *Configuration*.

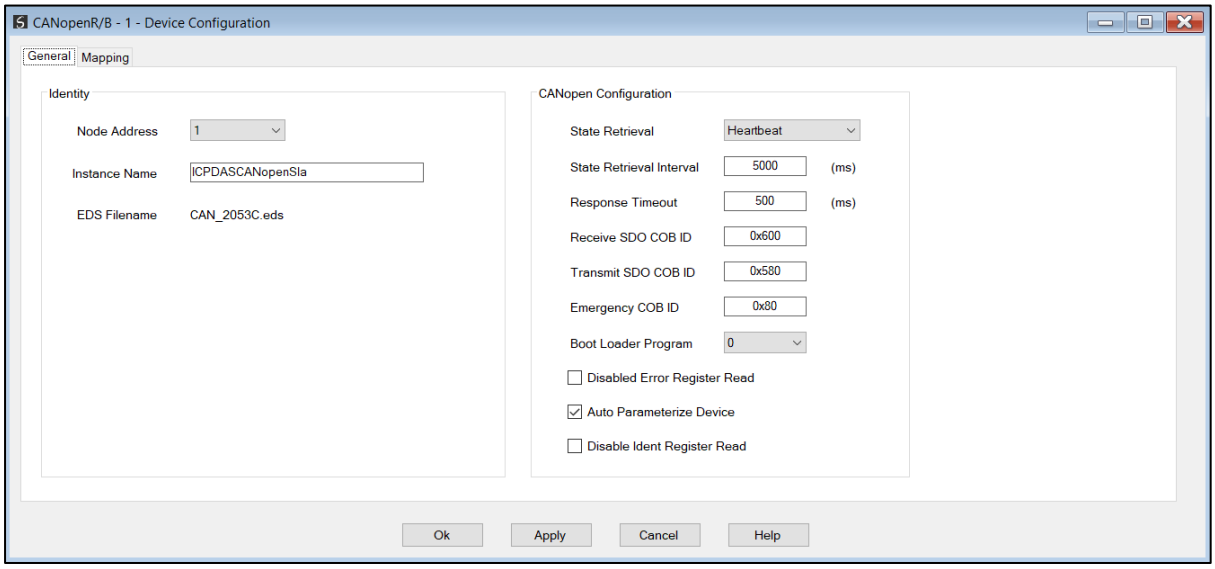


Figure 3.41 – Device General configuration parameters

The General configuration consists of the following parameters:

Parameter	Description
Node Address	The node address of the CANopen Slave device on the CANopen network.
Instance Name	The device instance name.
EDS Filename	Filename of the EDS file used for this slave device.
State Retrieval	This is the method used to retrieve the operational state of the Slave device. **Disabled** The CANopen Router will not attempt or expect the CANopen slave device to send its operational state. **Heartbeat** The CANopen slave device will automatically (without the need for a request from the Master) send its operational state. **NOTE:** In many field devices the Heartbeat has been disabled by default. If the *Auto Parameterize Device* option has **not** been selected, the user will need to go to the device parameters (see *Diagnostics* section) and

	change the *Producer Heartbeat Time* (parameter 1017) to a non-zero value. **Guarding** The CANopen Router will manually request the operational state of the device at the *State Retrieval Interval.*
State Retrieval Interval	When using the State Retrieval method *Guarding*, this parameter will be the interval at which the CANopen Router will manually retrieve the operational state of the CANopen slave device.
Response Timeout	The amount of time the CANopen Router will wait before it will flag that the CANopen slave device did not respond to a request from the CANopen Router.
Received SDO COB ID	This is the base COB ID used for receiving SDO messages from the CANopen Slave device. **NOTE:** This value will be populated by Slate and it is recommended that the user does not change the default value.
Transmit SDO COB ID	This is the base COB ID used for transmitting SDO messages from the CANopen Slave device. **NOTE:** This value will be populated by Slate and it is recommended that the user does not change the default value.
Emergency COB ID	This is the base COB ID used for transmitting Emergency (EMCY) messages from the CANopen Slave device. **NOTE:** This value will be populated by Slate and it is recommended that the user does not change the default value.
Boot Loader Program	When using the sub-sea profile (CiA 443), this will allow the CANopen Router to automatically set the bootloader program to be used by the CANopen Slave on start-up.
Disabled Error Register Read	When this is parameter is set, the CANopen Router in Master mode will not attempt to read the Error Register (0x1001) of the Slave device during connection establishment.
Auto Parameterize Device	The CANopen Router/B module will, in CANopen Master Mode, configure certain parameters in the CANopen Slave device automatically during connection establishment. When this parameter is set, the CANopen Router/B will setup the Heartbeat, TPDOs, and RPDOs based on the module configuration in Slate.

Disable Ident Register Read	When this is parameter is set, the CANopen Router in Master mode will not attempt to read the Ident Register (0x1018) of the Slave device during connection establishment.

Table 3.12 – Device General configuration parameters

3.5.5. CANopen Slave Device - PDO Mapping

The module can be configured to exchange process data (via the CANopen Process Data Objects – PDOs) between the various CANopen Slave devices and the primary interface (EtherNet/IP Target, Modbus Slave, Modbus Master, or EtherNet/IP Originator).

When the primary interface is EtherNet/IP Target, the CANopen Router will allow the user to exchange the CANopen device Process Data using the following methods:

- Allen-Bradley Logix Tag reads and writes (using Direct-to-Tag).
- EtherNet/IP Class 1 Target connection. The last two (of the four) class 1 connections can be used to provide CANopen Device data to the PLC / Controller.

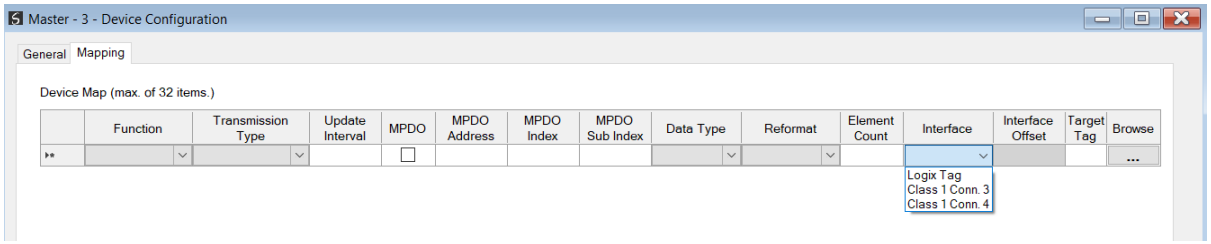


Figure 3.42 – EtherNet/IP Target Mode - Device Mapping parameters

When the primary interface is Modbus Slave or Modbus Master, the CANopen Router will allow the user to read data from a CANopen Slave device into a configurable Modbus Register and/or write data to a CANopen Slave device from a configurable Modbus Register. The internal Modbus registers can then be exchanged with a remote Modbus Master (when operating as a Modbus Slave) or with remote Modbus Slave devices using the Modbus Auxiliary Map (when operating as a Modbus Master).

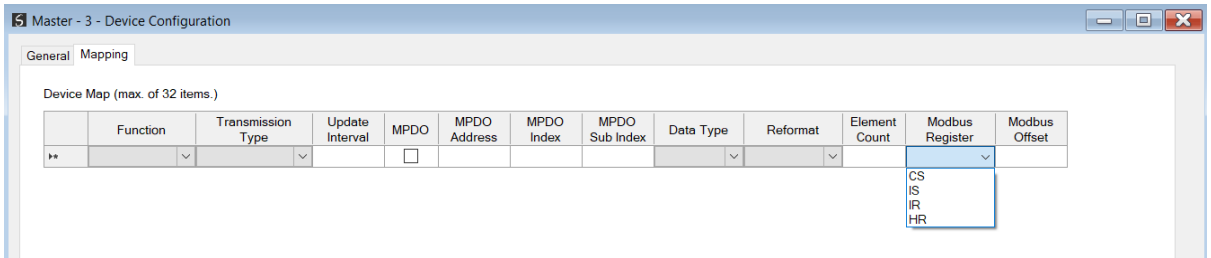


Figure 3.43 – Modbus Master/Slave Mode - Device Mapping parameters

When the primary interface is EtherNet/IP Originator, the CANopen Router will allow the user to exchange the CANopen device Process Data using the following methods:

- Class 1 connection Input and Output Assemblies from the configured EtherNet/IP devices.
- Explicit Messaging Data Map – which is populated with the data from the EtherNet/IP Explicit Message Mapping.

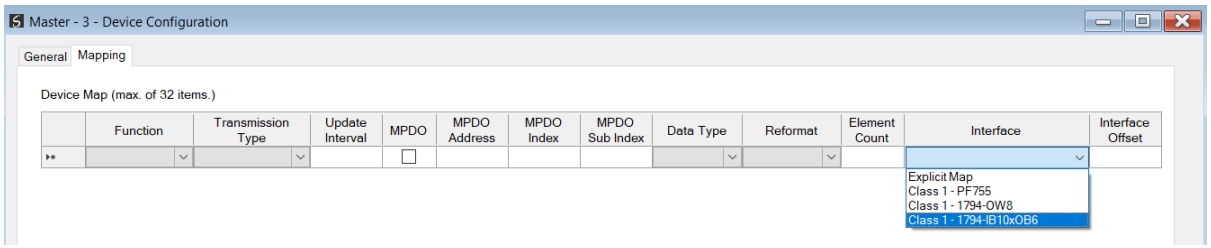


Figure 3.44 – EtherNet/IP Originator - Device Mapping parameters

Below are the common parameters (between each of the primary interfaces) for each PDO mapping item.

Parameter	Description
Function	There are two functions supported for mapping field device PDOs (process variables). **TPDO x (Rx)** TPDOs are the PDOs **received from** the CANopen Slave device. A total of four TPDOs can be used if multiplexing is not used (see MPDO section). Each non-MPDO PDO received from the Slave device can be up to 8 bytes (e.g. two 32-bit Reals). If MPDO is used a max of 4 bytes can be exchanged. **RPDO x (Tx)** RPDOs are the PDOs **sent to** the CANopen Slave device. A total of four RPDOs can be used if multiplexing is not used (see MPDO section). Each non-MPDO PDO sent to the Slave device can be up to 8 bytes (e.g. two 32-bit Reals). If MPDO is used a max of 4 bytes can be exchanged.
Transmission Type	**Sync** For RPDOs, the CANopen Router in Master mode will send out the PDO data to the CANopen slave after the SYNC packet. For TPDOs (if enabled in the Slave device), the CANopen slave device will send the PDO data after receiving the SYNC packet. **RemoteTxReq (TPDO-Rx only)** The CANopen Router will request the PDO from the CANopen slave at the update interval. **Evt-Timer**

	The CANopen Router in Master mode will send out the PDO data to the CANopen slave every Update Interval. If configured in the Slave, the PDO data will be received from the CANopen Slave at the configured interval in the CANopen Slave. **Evt-Manufacturer (TPDO-Rx only)** This is set if the CANopen Slave sends PDO data based on a specific event (e.g. Button Pushed). **Evt-Interface (RPDO-Tx only)** The CANopen Router in Master mode will send out the PDO data to the CANopen slave every time the relevant PDO bit is toggled in the CANopen Device Slave Trigger by the Primary Interface (e.g. Logix)
Update Interval	The time (in milliseconds) at which the PDOs will be requested (when transmission type is ***RemoteTxReq***) or at which the PDOs will be sent (when transmission type is ***Evt-Timer***).
MPDO	Each PDO can be multiplexed (if supported by the slave device) to have multiple process variables associated with it. With normal PDOs each PDO has a maximum of 8 bytes while with multiplexed PDOs each multiplexed process variable has maximum of 4 bytes. To enable Multiplexing the user must select the MPDO checkbox in the mapping of the PDO. **MPDO Address** The address of the process variable in the PDO. **MPDO Index** The index of the process variable in the PDO. **MPDO Sub Index** The sub index of the process variable in the PDO.
Data Type	The data type to be used when copying to/from the Primary Interface.
Reformat	How the data is reformatted before writing to, or after reading from, the CANopen slave. **None** – No reformatting will be done. **BB AA** – 16bit Byte Pair Swap **BB AA DD CC** – 32bit Byte Pair Swap **CC DD AA BB** – Word Swap **DD CC BB AA** – Word and Byte Pair Swap
Element Count	The number of elements to be used for the specific PDO. For example, the user can have 2 x 32-bit real values or 8 x 8-bit integers.  **NOTE:** The element count must be such that the total byte count (element count multiplied by the data type size) must not exceed: **8 bytes** for **non-multiplexed** mapped items, and **4 bytes** for **multiplexed** map items.

Interface	The interface selection will be used to provide the source or destination of the CANopen PDO data based on the Primary Interface selected: **EtherNet/IP Target** The user can select either a Logix tag or the input/output assemblies from the Logix Class 1 connection (*Class 1 Conn.3* or *Class 1 Conn.4*) **Modbus Slave/Master** Modbus Register (HR, IR, IS, CS) and the Offset in the Modbus Register. **EtherNet/IP Originator** The user can select either the internal Explicit Map (where the Explicit EtherNet/IP data exchange resides) or one of the Class 1 EtherNet/IP originator connections that have been added to the CANopen Router module.

Table 3.13 –Device Mapping parameters

3.5.5.1. ETHERNET/IP TARGET

When using the EtherNet/IP target as the primary interface, the user can select from the PDO mapping to either exchange data with the primary interface using a Logix Tag or Class 1 Connection 3 or Class 1 Connection 4.

A. LOGIX TAG

When using the Logix Tag Interface in the PDO mapping, the TPDO data from the CANopen Slave device will be written into the Target Tag specified in the mapping, and the RPDO data sent to the CANopen Slave device will be read from the Target Tag specified in the mapping.

The user will first need to ensure that a Logix controller has been selected in the general configuration. This can be done by either manually typing in the Logix path or browsing to the Logix controller.

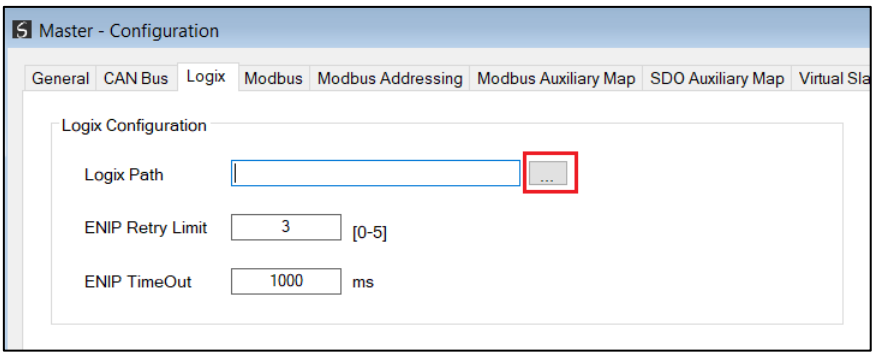


Figure 3.45 – EtherNet/IP Target – Logix Controller Browsing

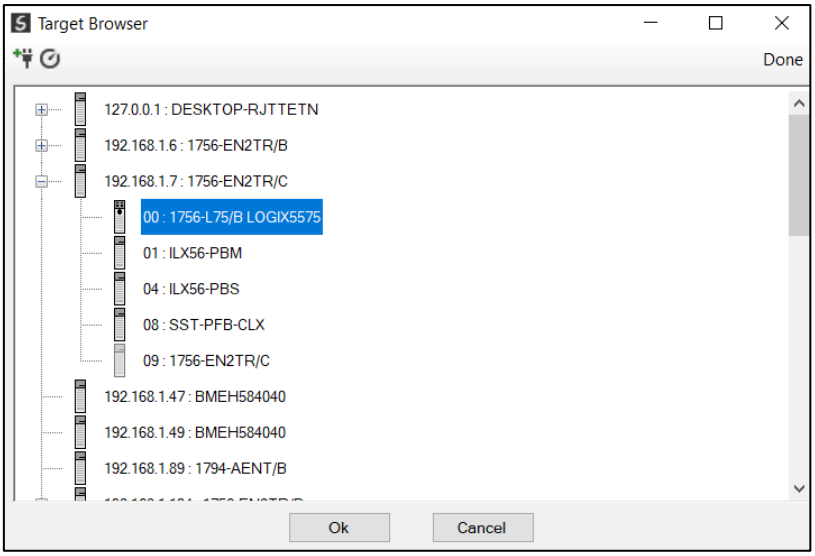


Figure 3.46 – EtherNet/IP Target – Logix Controller Selection

Next the user can either type in the Target Tag or Browse the Logix Controller for its tags.

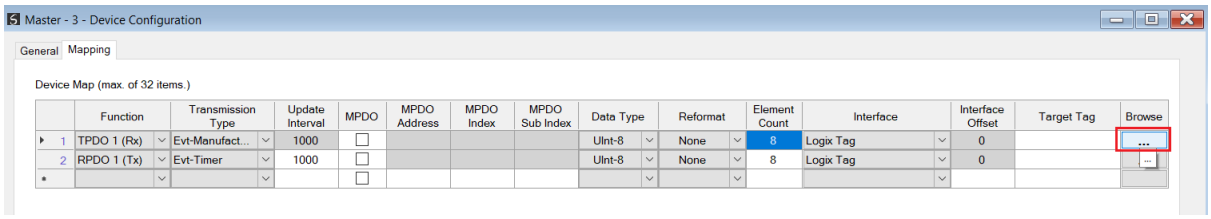


Figure 3.47 – EtherNet/IP Target – Logix Tags Browsing

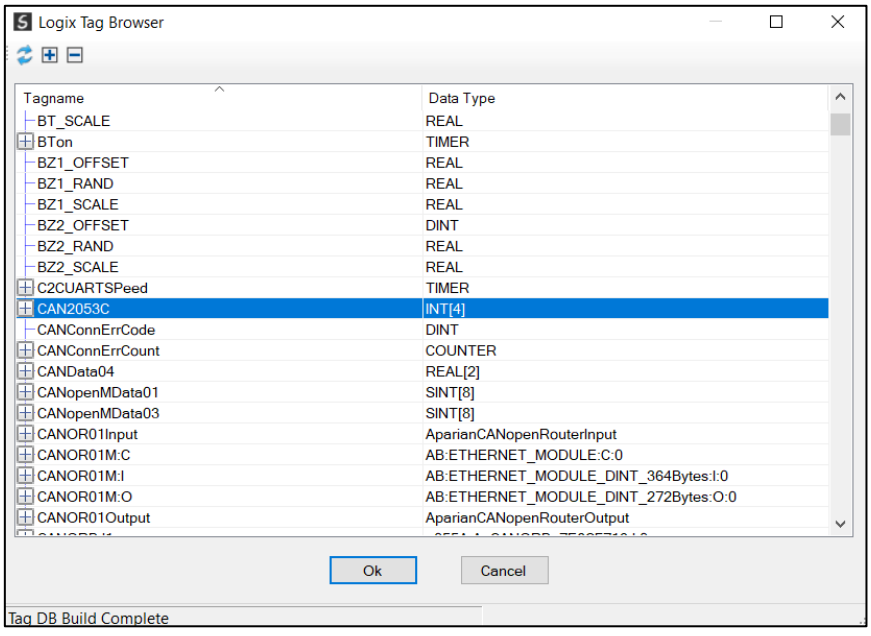


Figure 3.48 – EtherNet/IP Target – Logix Tags Selection

NOTE: The user must ensure that the selected Logix tag is sufficiently large to accommodate the specified PDO. For example, if the PDO returns two REAL values, the Logix Target Tag would need to be an array (minimum length of 2) rather than a single REAL.

B. CLASS 1 CONNECTION #3 / CLASS 1 CONNECTION #4

When using the Class 1 Connection 3 or Class 1 Connection 4 Interface in the PDO mapping, the TPDO data from the CANopen Slave device will be written into the Input Assembly of the module Class 1 connection, and the RPDO data sent to the CANopen Slave device will be read from the Output Assembly of the module Class 1 connection.

NOTE: The CANopen Router/B module uses 4 Class 1 Cyclic EtherNet/IP connections. The PDO data will be exchanged with either Connection 3 or Connection 4.

3.5.5.2. MODBUS MASTER / MODBUS SLAVE

When Modbus Master or Modbus Slave has been selected as the primary interface, the process variables (TPDOs) from the CANopen Slave device will be stored in the configured Modbus Register (Holding Register, Input Register, Input Status, or Coil Status) with the specified offset. The process variable (RPDOs) that will be sent to the CANopen Slave device will be read from the configured Modbus Register (Holding Register, Input Register, Input Status, or Coil Status) with the specified offset.

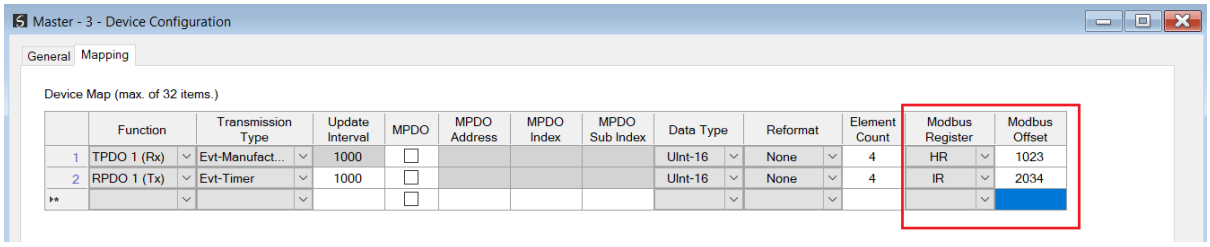


Figure 3.49 – Modbus – Register and Offset Selection

NOTE: The user will need to ensure that when writing to the CANopen Router/B Modbus Registers, that the Modbus Registers being used for CANopen PDO data are not inadvertently overwritten.

3.5.5.3. ETHERNET/IP ORIGINATOR INTERFACE

When using the EtherNet/IP Originator as the primary interface, the user can select the PDO mapping to either exchange data with the primary interface using the Explicit Data Map or Class 1 Connections from the devices configured in the EtherNet/IP Connections in Slate.

A. EXPLICIT MAP

When using the Explicit Map for PDO data exchange, the data will be written to, and read from the Explicit Map at the specified offset. The PDO data can then be accessed from the EtherNet/IP Devices in the Explicit EtherNet/IP Mapping in the general configuration.

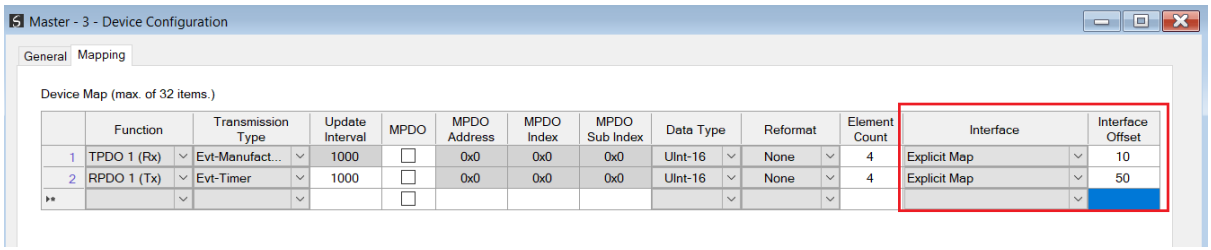


Figure 3.50 – EtherNet/IP Originator – Explicit Map and offset

B. ETHERNET/IP CLASS 1 ORIGINATOR CONNECTIONS

When using the Class 1 Connection Originated connections to exchange data with the CANopen device PDOs, the user will need to select a Device from the Interface drop down list which will match the configured devices in the *EtherNet/IP Connections* in the module configuration (as shown below).

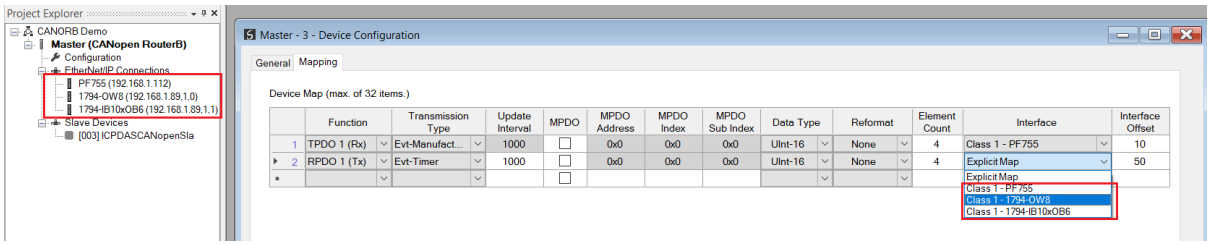


Figure 3.51 – EtherNet/IP Originator – Device Selection

NOTE: When using TPDO (Rx) with a EtherNet/IP device from the EtherNet/IP Connections list, the data will be written to the output assembly (from originator to target) of the Class 1 connection of the EtherNet/IP device. When using RPDO (Tx) with a EtherNet/IP device from the EtherNet/IP Connections list, the data will be read from the input assembly (from target to originator) of the Class 1 connection of the EtherNet/IP device.

3.5.6. CANOPEN SLAVE DEVICE - PARAMETERIZATION

Each field device provides a range of parameters that can be accessed using the SDO communication parameters of the CANopen Slave device. This will allow the user to view and (with certain parameters) change the settings in the slave device. To access the slave device

parameters the user will need to open the *Status* window of the slave device and select the parameters tab (as shown below):

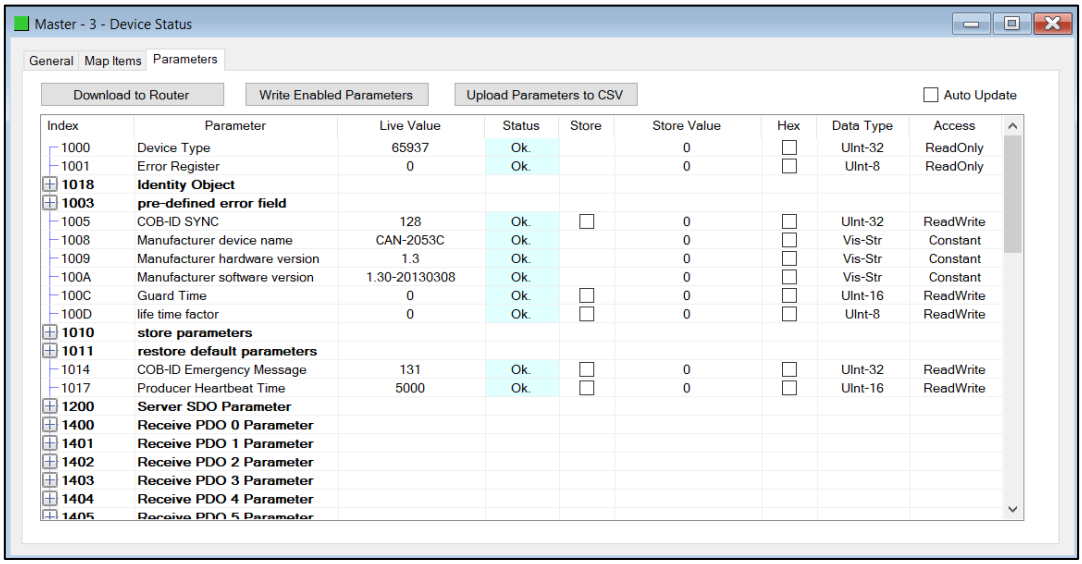


Figure 3.52 – Slave device parameters

These parameters will be listed from the EDS file used to instantiate the CANopen Slave device. See the *Device Parameterization* section in the *Operations* section.

3.6. CANOPEN SLAVE MODE

The module can be configured to operate as a CANopen Slave on the CANopen network (see the *General Configuration*). The module can emulate up to 125 PDOs from different CANopen node addresses when operating as a Slave on the CANopen network and exchange process data (via the CANopen Process Data Objects – PDOs) between the CANopen Mater and the primary interface (EtherNet/IP Target, Modbus Slave, Modbus Master, or EtherNet/IP Originator).

3.6.1. VIRTUAL DEVICE MAP

The mapping for the CANopen Router when operating as a CANopen Slave will be done through the Virtual Device Map (as shown below).

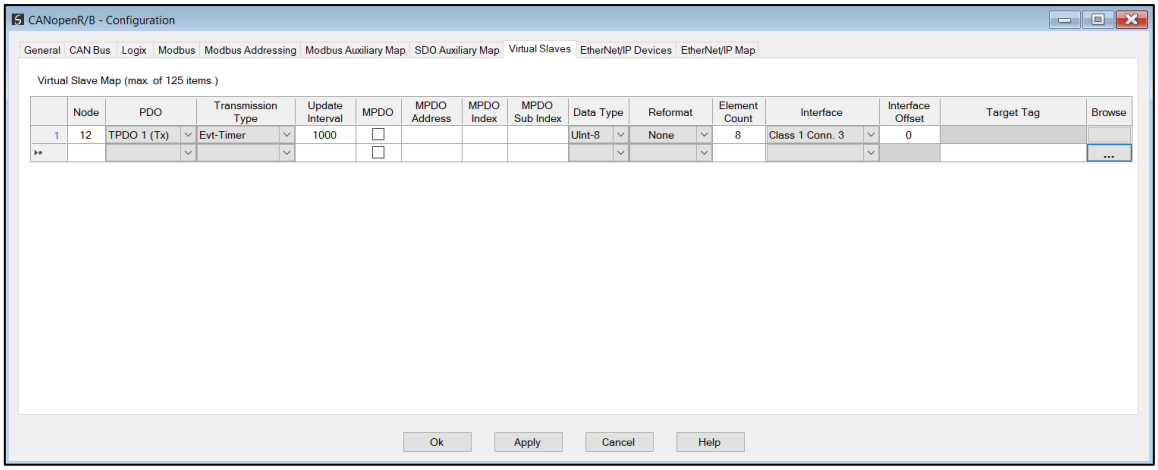


Figure 3.53 – CANopen Router/B as Slave – PDO Mapping

When the primary interface is EtherNet/IP Target, the CANopen Router will allow the user to exchange the CANopen device Process Data using the following methods:

- Allen-Bradley Logix Tag reads and writes (using Direct-to-Tag).
- EtherNet/IP Class 1 Target connection. The last two (of the four) class 1 connections can be used to provide CANopen Device data to the PLC / Controller.

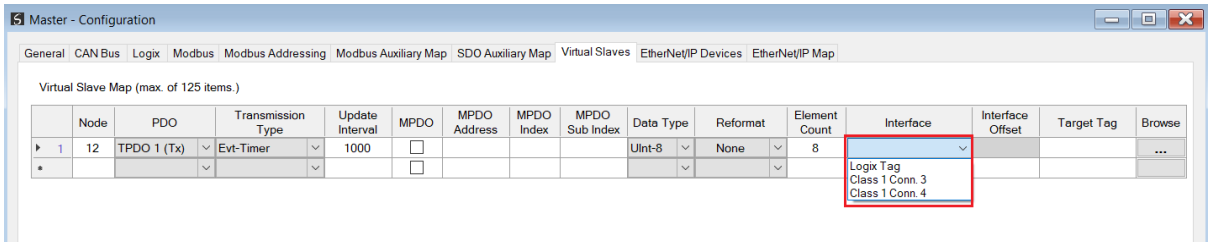


Figure 3.54 – EtherNet/IP Target Mode – Virtual Slave Map

When the primary interface is Modbus Slave or Modbus Master, the CANopen Router will receive the data from a CANopen Master and write into a configurable Modbus Register and/or send data to a CANopen Master from a configurable Modbus Register. The internal Modbus registers can then be exchanged with a remote Modbus Master (when operating as a Modbus Slave) or with remote Modbus Slave devices using the Modbus Auxiliary Map (when operating as a Modbus Master).

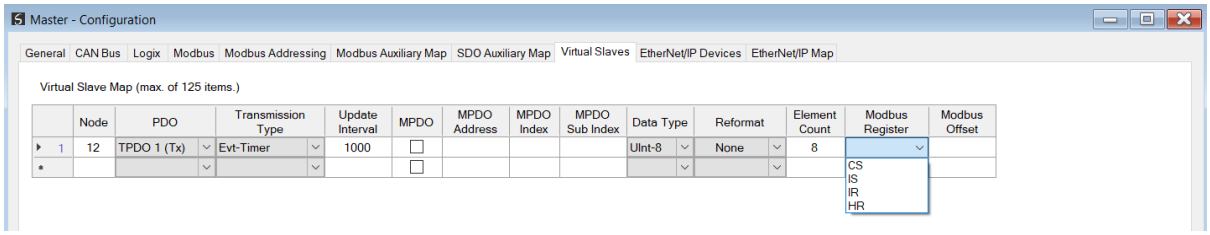


Figure 3.55 – Modbus Master/Slave Mode - Virtual Slave Map

When the primary interface is EtherNet/IP Originator, the CANopen Router will allow the user to exchange the CANopen device Process Data using the following methods:

- Class 1 connection Input and Output Assemblies from the configured EtherNet/IP devices.
- Explicit Messaging Data Map – which is populated with the data from the EtherNet/IP Explicit Message Mapping.

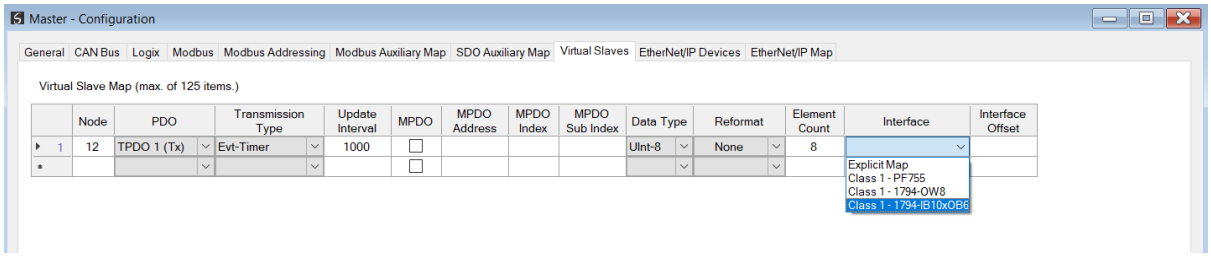


Figure 3.56 – EtherNet/IP Originator - Virtual Slave Map

Below are the common parameters (between each of the primary interfaces) for each Virtual Slave PDO mapping item.

Parameter	Description
Node	The CANopen Slave node address the CANopen Router will emulate. The module can emulate up to 125 slaves.
PDO	There are two functions supported for mapping PDOs (process variables) for the CANopen Router when operating as a CANopen slave. **TPDO x (Tx)** TPDOs are the PDOs **sent to** the remote CANopen Master. **RPDO x (Rx)** RPDOs are the PDOs **received from** the remote CANopen Master.
Transmission Type	**Sync** The CANopen Router in Slave mode will send out the PDO data to the CANopen Master once a ***SYNC*** packet has been received. **RemoteTxReq (TPDO only)** The CANopen Router will transmit the PDO to the CANopen Master when requested from the CANopen Master. **Evt-Timer (TPDO only)** The CANopen Router in Slave mode will send out the PDO data to the CANopen Master every Update Interval. **Evt-Interface (TPDO only)** The CANopen Router in Slave mode will send out the PDO data to the CANopen Master every time the relevant PDO bit is toggled in the Slave Triggers by the configured Primary Interface.

	Evt-Manufacturer (RPDO only) This is set if the CANopen Master sends PDO data based on a specific event (e.g. Timer).
Update Interval	The time (in milliseconds) at which the PDOs will be sent (when transmission type is ***Evt-Timer*** or ***Remote Tx Request***).
MPDO	Each PDO can be multiplexed (if supported by the CANopen Master) to have multiple process variables associated with it. With normal PDOs each PDO has a maximum of 8 bytes while with multiplexed PDOs each multiplexed process variable has maximum of 4 bytes. To enable Multiplexing the user must select the MPDO checkbox in the mapping of the PDO. **MPDO Address** The address of the process variable in the PDO. **MPDO Index** The index of the process variable in the PDO. **MPDO Sub Index** The sub index of the process variable in the PDO.
Data Type	The data type to be used when copying to/from the Primary Interface.
Reformat	How the data is formatted before reading or writing from/to the CANopen slave. **None** – No reformatting will be done. **BB AA** – 16bit Byte Pair Swap **BB AA DD CC** – 32bit Byte Pair Swap **CC DD AA BB** – Word Swap **DD CC BB AA** – Word and Byte Pair Swap
Element Count	The number of elements to be used for the specific PDO. For example, the user can have 2 x 32-bit real values or 8 x 8-bit integers.  **NOTE:** The element count must be such that the total byte count (element count multiplied by the data type size) must not exceed: **8 bytes** for **non-multiplexed** mapped items, and **4 bytes** for **multiplexed** map items.
Interface	The interface selection will be used to provide the source or destination of the CANopen PDO data based on the Primary Interface selected: **EtherNet/IP Target** The user can select either a Logix tag or the input/output assemblies from the Logix Class 1 connection (*Class 1 Conn.3* or *Class 1 Conn.4*) **Modbus Slave/Master** Modbus Register (HR, IR, IS, CS) and the Offset in the Modbus Register. **EtherNet/IP Originator**

	The user can select either the internal Explicit Mag (where the Explicit EtherNet/IP data exchange resides) or one of the Class 1 EtherNet/IP originator connections that have been added to the CANopen Router module.

Table 3.14 – Virtual Slave Map parameters

3.6.1.1. ETHERNET/IP TARGET

When using the EtherNet/IP target as the primary interface, the user can select from the PDO mapping to exchange data with the primary interface using either Logix Tag, Class 1 Connection 3 or Class 1 Connection 4.

A. LOGIX TAG

When using the Logix Tag Interface in the PDO mapping, the RPDO (Rx) data from the CANopen Master will be written into the Target Tag specified in the mapping, and the TPDO (Tx) data sent to the CANopen Master will be read from the Target Tag specified in the mapping.

The user will first need to ensure that a Logix controller has been selected in the general configuration. This can be done by either manually typing in the Logix path or browsing to the Logix controller.

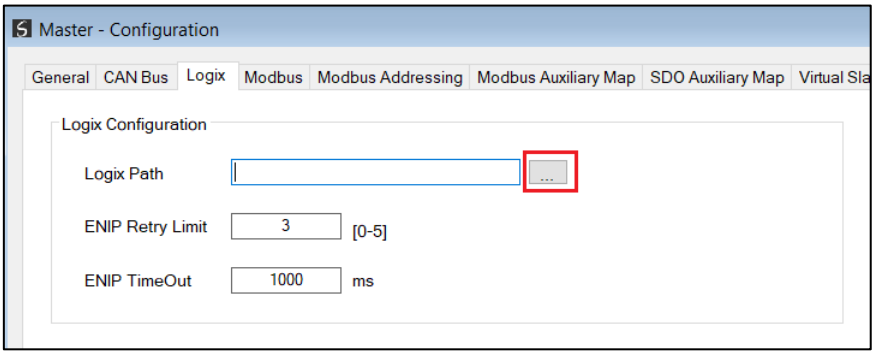


Figure 3.57 – EtherNet/IP Target – Logix Controller Browsing

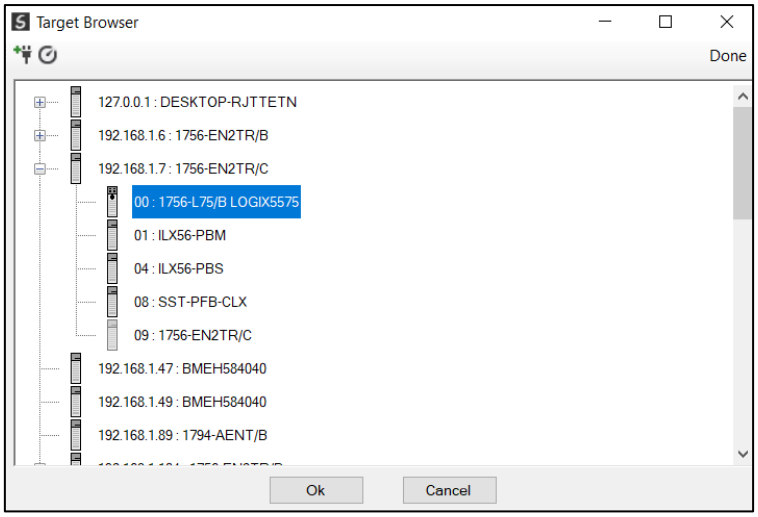


Figure 3.58 – EtherNet/IP Target – Logix Controller Selection

Next the user can either type in the Target Tag or Browse the Logix Controller for its tags.

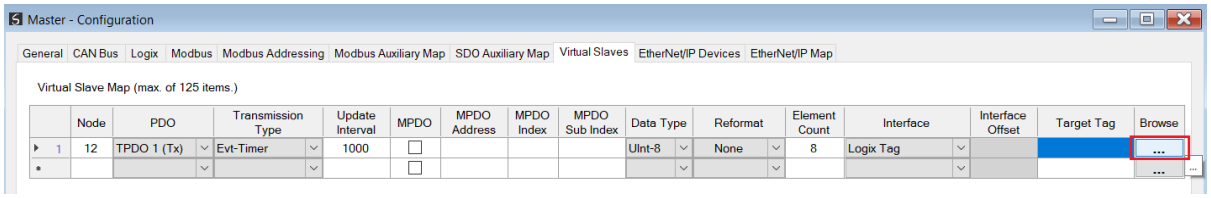


Figure 3.59 – EtherNet/IP Target – Logix Tags Browsing

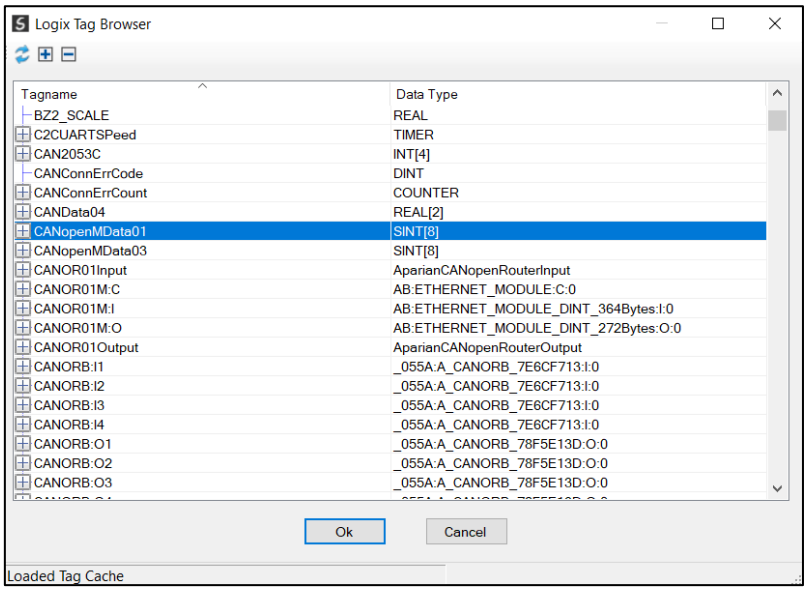


Figure 3.60 – EtherNet/IP Target – Logix Tags Selection

NOTE: The user must ensure that the selected Logix tag is sufficiently large to accommodate the specified PDO. For example, if the PDO returns two REAL values, the Logix Target Tag would need to be an array (minimum length of 2) rather than a single REAL.

B. CLASS 1 CONNECTION #3 / CLASS 1 CONNECTION #4

When using the Class 1 Connection 3 or Class 1 Connection 4 Interface in the PDO mapping, the RPDO (Rx) data from the CANopen Master will be written into the Input Assembly of the module Class 1 connection, and the TPDO (Tx) data sent to the CANopen Master will be read from the Output Assembly of the module Class 1 connection.

NOTE: The CANopen Router/B module uses 4 Class 1 Cyclic EtherNet/IP connections. The PDO data will be exchanged with either Connection 3 or Connection 4.

3.6.1.2. MODBUS MASTER / MODBUS SLAVE

When Modbus Master or Modbus Slave has been selected as the primary interface, the process variables (RPDOs) from the CANopen Master will be stored in the configured Modbus Register (Holding Register, Input Register, Input Status, or Coil Status) with the specified offset. The process variable (TPDOs) that will be sent to the CANopen Master will also be read from the Modbus Register (Holding Register, Input Register, Input Status, or Coil Status) with the specified offset.

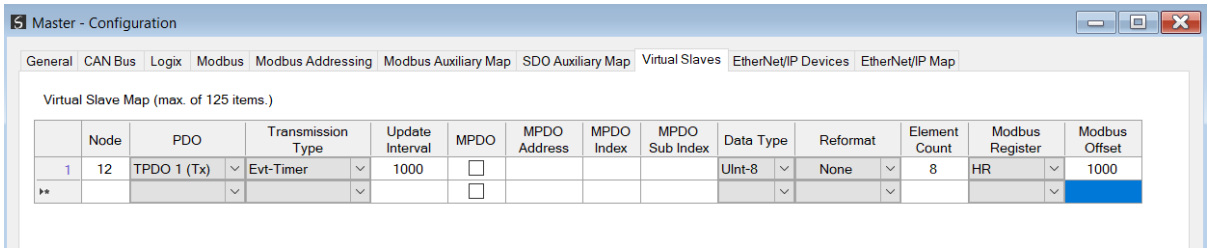


Figure 3.61 – Modbus – Register and Offset Selection

NOTE: The user will need to ensure that when writing to the CANopen Router/B Modbus Registers, that the Modbus Registers being used for CANopen PDO data are not inadvertently overwritten.

3.6.1.3. ETHERNET/IP ORIGINATOR INTERFACE

When using the EtherNet/IP Originator as the primary interface, the user can select the PDO mapping to either exchange data with the primary interface using the Explicit Data Map or Class 1 Connections from the devices configured in the EtherNet/IP Connections in Slate.

A. EXPLICIT MAP

When using the Explicit Map for PDO data exchange, the data will be written to, and read from, the Explicit Map at a configured offset. The PDO data can then be accessed from the EtherNet/IP Devices in the Explicit EtherNet/IP Mapping in the general configuration.

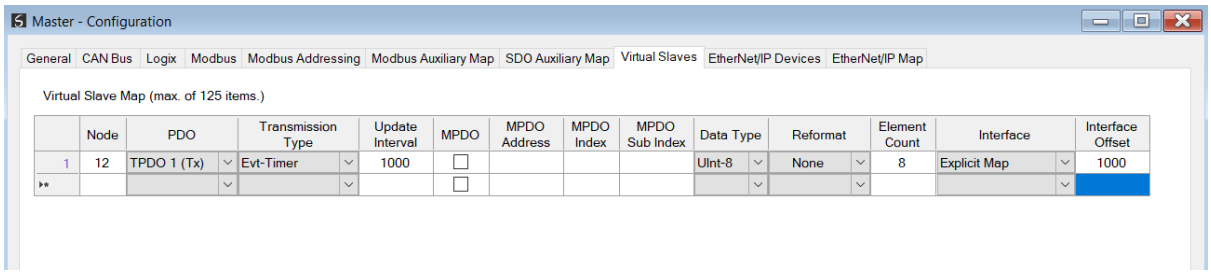


Figure 3.62 – EtherNet/IP Origiantor – Explicit Map and offset

B. ETHERNET/IP CLASS 1 ORIGINATOR CONNECTIONS

When using the Class 1 Connection Originated connections to exchange data with the CANopen device PDOs, the user will need to select a Device from the Interface drop down list which will match the configured devices in the *EtherNet/IP Connections* in the module configuration (as shown below).

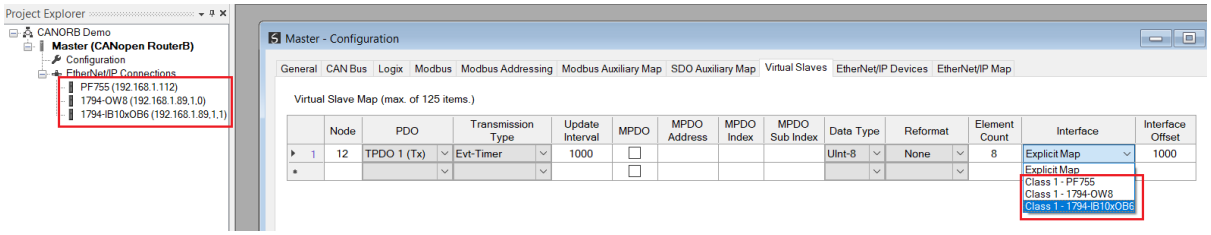


Figure 3.63 – EtherNet/IP Origiantor – Device Selection

NOTE: When using RPDO (Rx) with a EtherNet/IP device from the EtherNet/IP Connections list, the data will be written to the output assembly (from originator to target) of the Class 1 connection of the EtherNet/IP device. When using TPDO (Tx) with a EtherNet/IP device from the EtherNet/IP Connections list, the data will be read from the input assembly (from target to originator) of the Class 1 connection of the EtherNet/IP device.

3.7. ETHERNET/IP TARGET CONFIGURATION

3.7.1. CLASS 1 CONNECTION

3.7.1.1. ADD MODULE TO I/O CONFIGURATION

The CANopen Router/B can be easily integrated with Allen-Bradley Logix family of controllers. Integration with the Logix family in Studio5000 makes use of the EDS Add-On-Profile (AOP).

A. EDS AOP (LOGIX V21+)

Before the module can be added to the tree the module's EDS file must be registered.

Using RSLinx, the EDS file can be uploaded from the device after which the EDS Hardware Installation tool will be invoked to complete the registration.

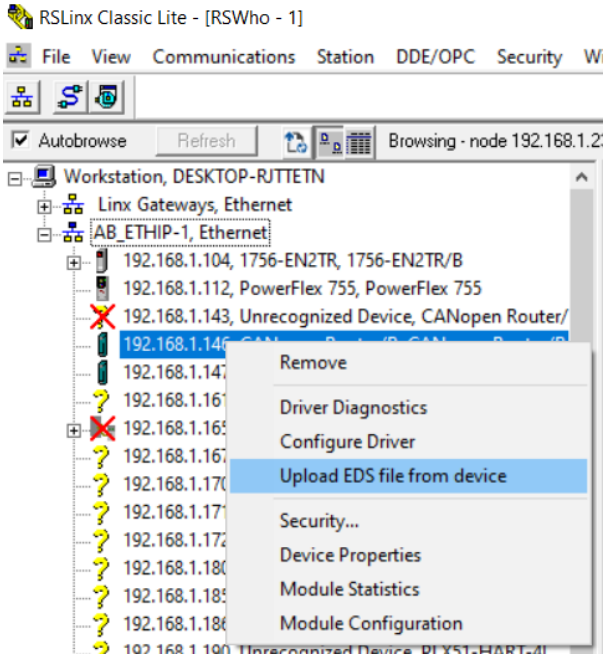


Figure 3.64 – EDS file upload from CANopen Router/B

Alternatively, the EDS file can be downloaded from the product web page at www.aparian.com and registered manually using the EDS Hardware Installation Tool shortcut under the Tools menu in Studio 5000.

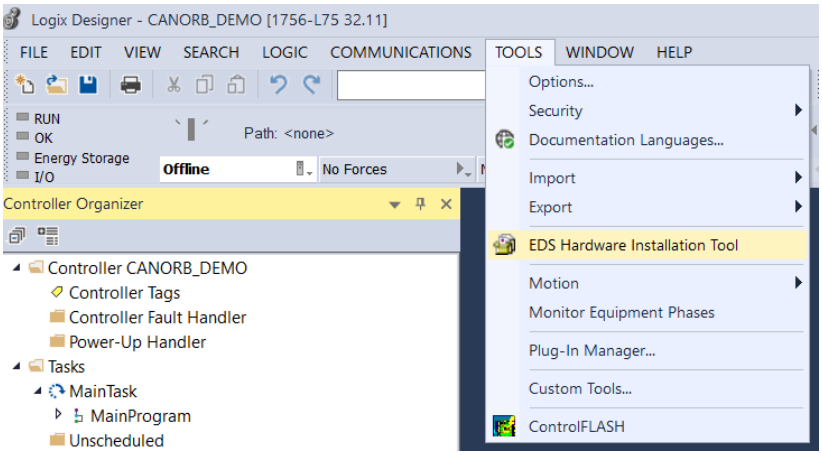


Figure 3.65 - EDS Hardware Installation Utility

After the EDS file has been registered, the module can be added to the Logix IO tree in Studio 5000. Under a suitable Ethernet bridge module in the tree, select the Ethernet network, right-click and select the New Module option.

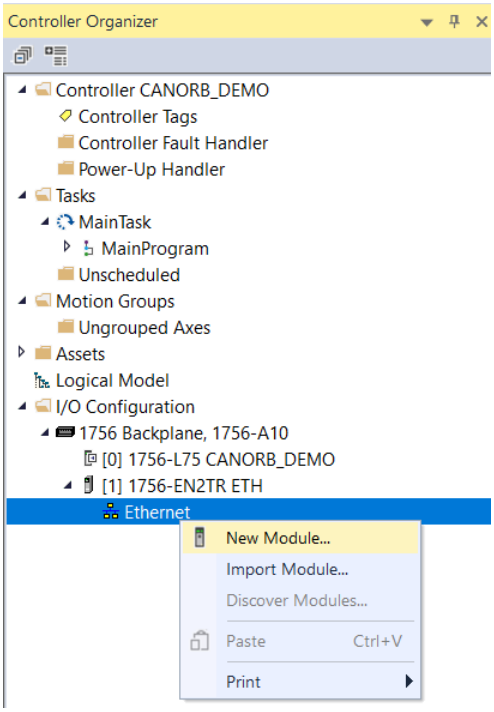


Figure 3.66 – Adding a module

The module selection dialog will open. To find the module more easily, use the Vendor filter to select only the Aparian modules as shown in the figure below.

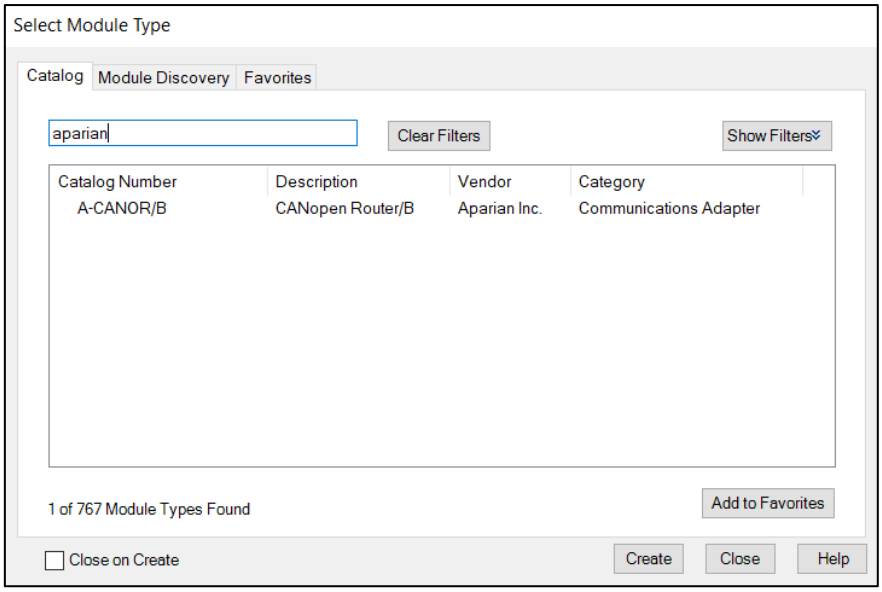


Figure 3.67 – Selecting the module

Locate and select the CANopen Router/B module and select the ***Create*** option. The module configuration dialog will open, where the user must specify the Name and IP address as a minimum to complete the instantiation.

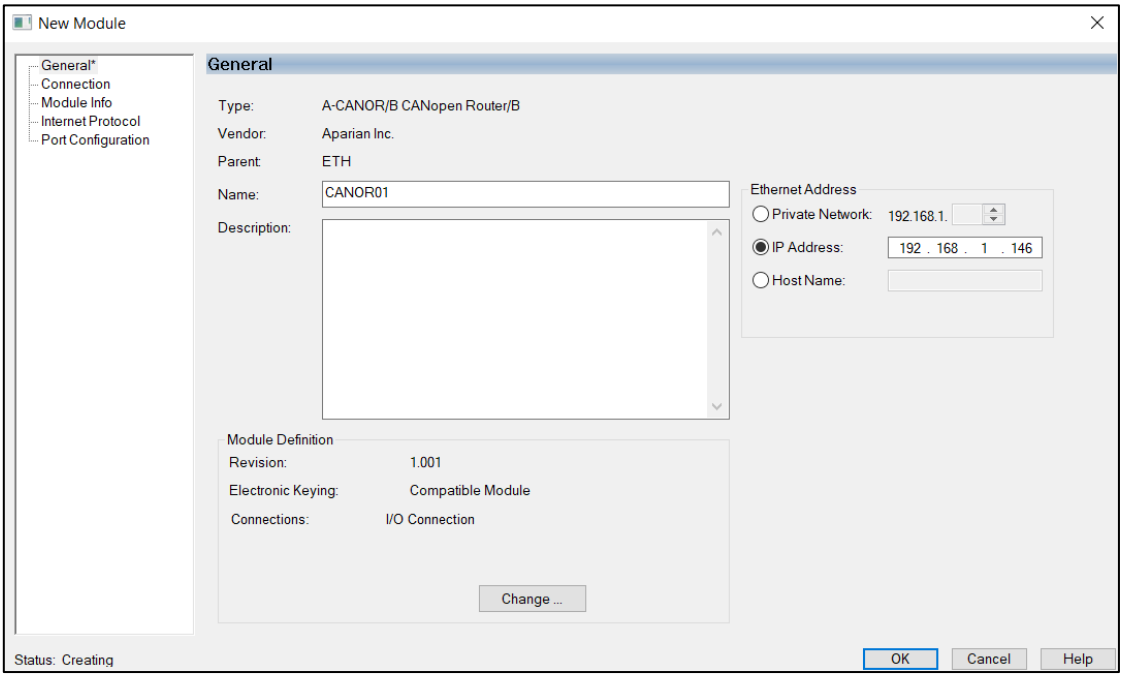


Figure 3.68 – Module instantiation

The user will need to ensure that all four connections have been enabled for the module. Once the instantiation is complete the module will appear in the Logix IO tree. Under the module definition, click the *Change* button.

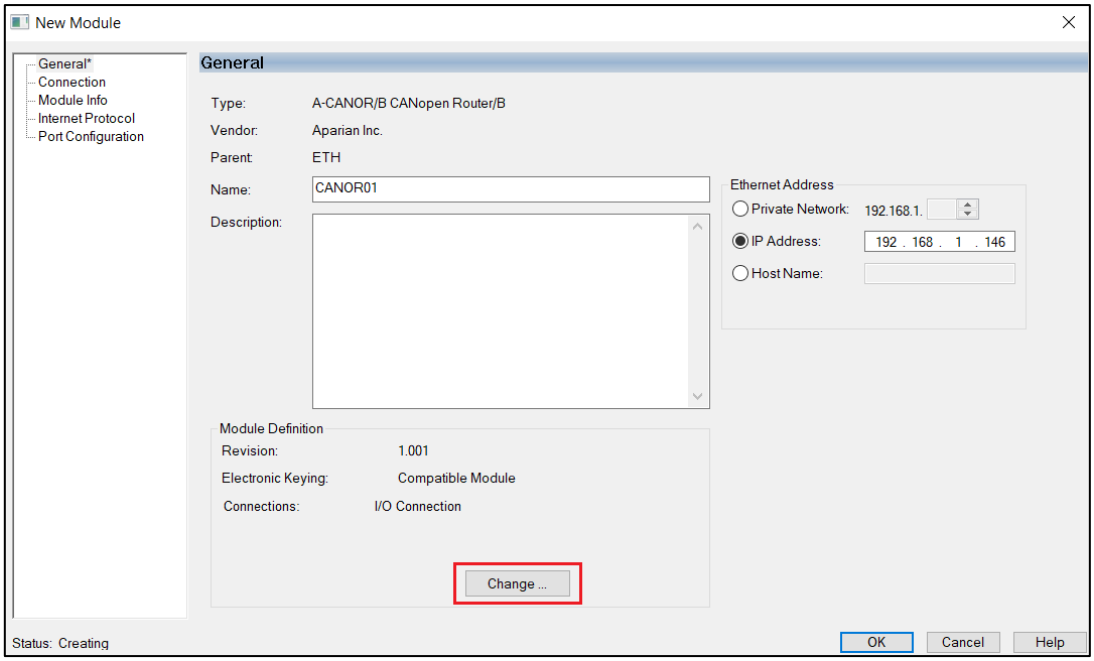


Figure 3.69 – Change number of IO Connections

Next the user will need to select each of the 4 connections available.

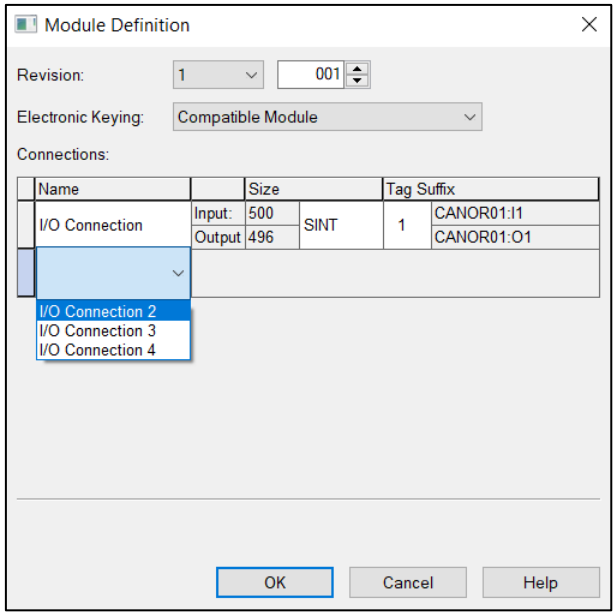


Figure 3.70 – Selection of IO Connections

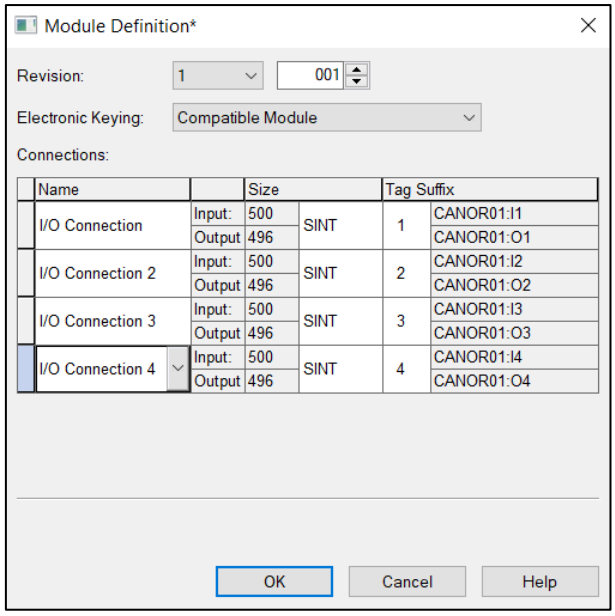


Figure 3.71 – All four connections selected.

Now the CANopen Router/B module will be in the Logix IO tree.

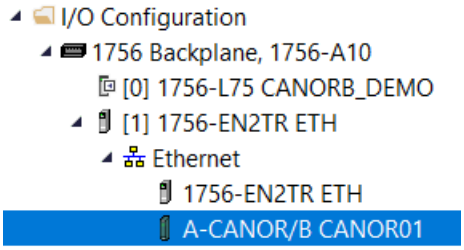


Figure 3.72 – Logix IO tree

The Module Defined Data Types will automatically be created during the instantiation process. These module defined tags will need to be copied to and from meaningful structures. This can be done by importing the example UDTs and Mapping Routines.

3.7.1.2. IMPORTING UDTS AND MAPPING ROUTINES

To simplify the mapping of the input image, a Logix 5000 Routine Partial Import (L5X) file is provided.

This file can be imported by right-clicking on the required Program and selecting the Import Routine option.

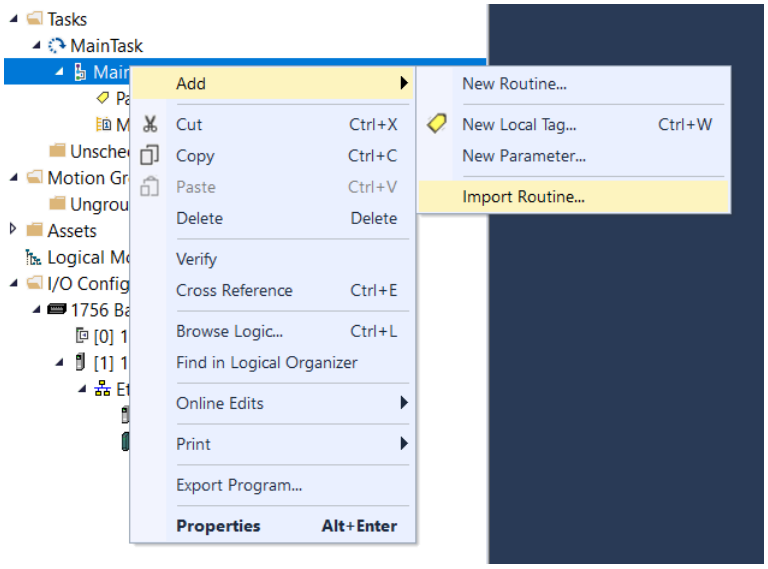


Figure 3.73 – Logix 5000 Importing CANopen Router/B specific routine and UDTs

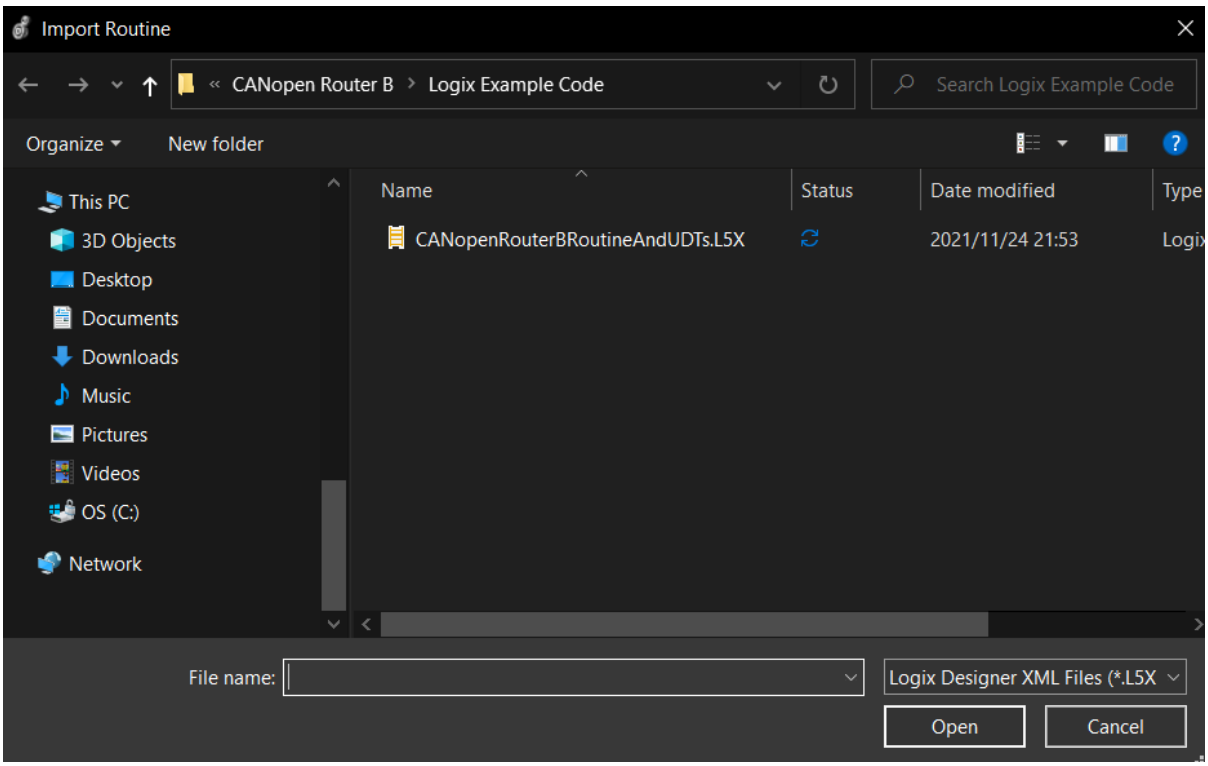


Figure 3.74 - Selecting partial import file

The import will create the following:

- The required UDTs (user defined data types)
- The controller tags representing the Input and Output assemblies for each of the four connections.
- A routine mapping the CANopen Router/B module to the aforementioned tags.

The user may need to change the routine to map to the correct CANopen Router module instance name, and make sure that the mapping routine is called by the Program's Main Routine.

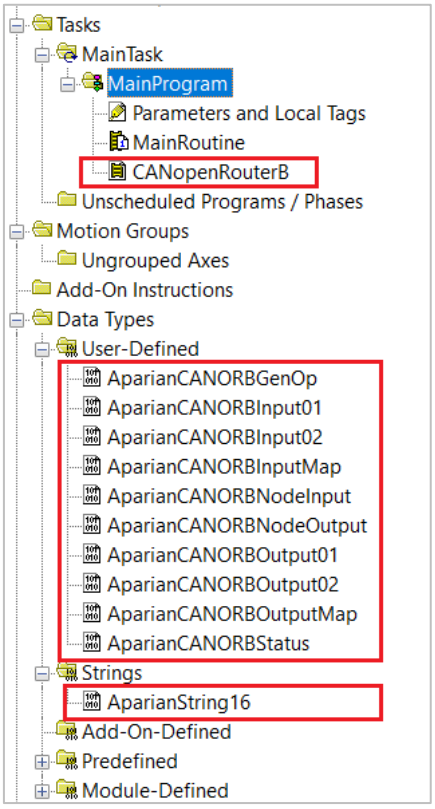


Figure 3.75 - Imported Studio 5000 objects

3.7.2. LOGIX TAG

The Logix Tag setup for EtherNet/IP Target Mode when the CANopen Router/B is operating as a CANopen Master is shown in section 3.5.5.1.a while the Logix Tag setup for EtherNet/IP Target Mode when the CANopen Router/B is operating as a CANopen Master is shown in section 3.6.1.1.a.

3.8. MODBUS MASTER CONFIGURATION

The CANopen Router/B can operate as a Modbus Master for Modbus TCP, Modbus RTU232, and Modbus RTU485 simultaneously. The user will need to configure the relevant Modbus Parameters as shown in section 3.4.4 followed by the configuration of the Modbus Auxiliary Map. This map will allow the user configure various read and write functions to external Modbus Registers to and from the internal Modbus registers.

See section 3.4.6 for more details on the various fields in the Modbus Auxiliary Map. The Modbus Aux Map will be executed in a linear manner and a mapped item will be executed at the *Update Rate* in the Modbus parameters in section 3.4.4.

3.9. MODBUS SLAVE CONFIGURATION

The CANopen Router/B can operate as a Modbus Slave for Modbus TCP, Modbus RTU232, and Modbus RTU485 simultaneously. The user will need to configure the relevant Modbus Parameters as shown in section 3.4.4.

The Modbus Node Number will need to be configured in section 3.4.4 to allow a Modbus Master to access the CANopen Router/B as a Modbus Slave device.

3.10. ETHERNET/IP ORIGINATOR CONFIGURATION

The CANopen Router/B can operate as a EtherNet/IP connection originator for cyclic (Class 1) or explicit (Class 3 or UCMM) data exchange. The explicit messaging can be setup in the *EtherNet/IP Devices* and *EtherNet/IP Map* in the Master configuration while the cyclic class 1 connections are added to the *EtherNet/IP Connections* node under the module in the Slate project tree.

3.10.1. EXPLICIT ETHERNET/IP MESSAGING

Up to five EtherNet/IP devices can be added for explicit messaging. The user will need to add each device as explained in section 3.4.9. Once the EtherNet/IP devices have been added the user can then configure the require mapping for the EtherNet/IP Explicit messaging as shown in section 3.4.10.

3.10.2. CYCLIC CLASS 1 CONNECTION

The CANopen Router/B can establish up to 5 cyclic Class 1 EtherNet/IP connections to EtherNet/IP devices. This can be done by either manually entering the connection data into the Connection Parameter window, or by importing the configuration from one or more of the following sources:

- Online Logix Controller
- Logix Controller L5X
- EDS File
- Connection Library

3.10.2.1. MANUAL CONFIGURATION

A class 1 connection can be added to the *EtherNet/IP Connections* tree by right-clicking on the tree in Slate and selecting *Add EtherNet/IP Connection.*

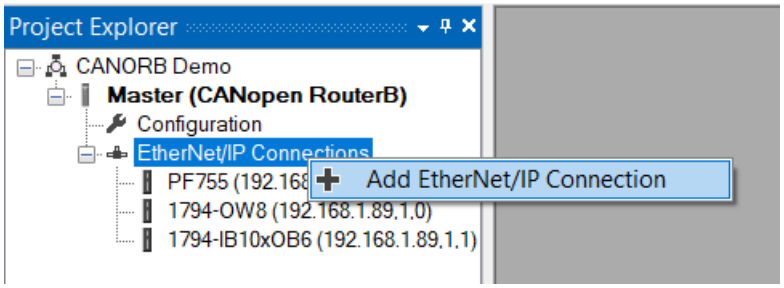


Figure 3.76 – Adding EtherNet/IP Class 1 Connection

Next the user will need to enter the connection parameters for the Class 1 connection.

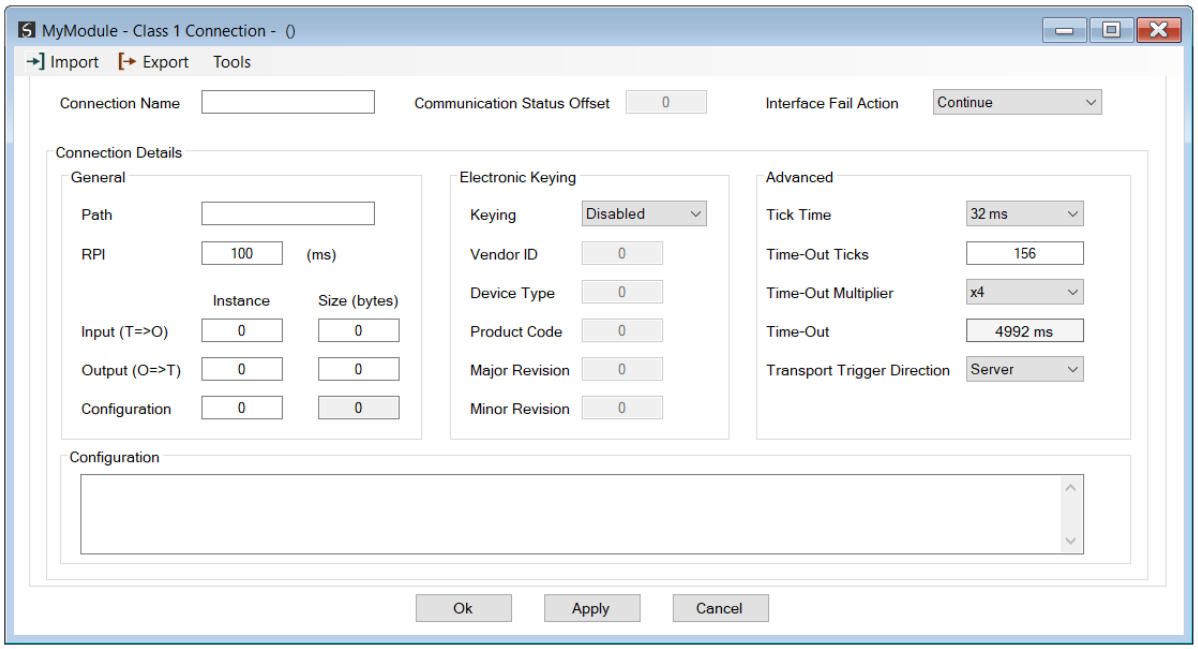


Figure 3.77 – EtherNet/IP Class 1 Connection Parameters

NOTE: It is recommended that the user not change the values in the *Advanced* frame of the connection parameters.

Parameter	Description
Connection Name	The instance name given to the Class 1 Connection.

Comm Status Offset	This is the offset in the data table (used to map EtherNet/IP device data) which provides the communication status of each EtherNet/IP device. The Communication Status is as shown below: Bit 0 - (1: Device online / 0:Device offline) Bit 1 to 7 – Reserved.
Interface Fail Action	When the CANopen Router is operating as a CANopen Slave and the CANopen Slave communication has timed-out, the EtherNet/IP IO can be configured to either keep the connection running as is or to change the connection status to program mode. This will allow the EtherNet/IP IO to go into a pre-determined state when the communication to the CANopen Master is lost. **NOTE: This is only valid when the module is operating as a CANopen Slave.**
General	
Path	The path to the target device. If the device is an Ethernet device then this will just be the IP address of the module. If the device is, for example, a module in a backplane or via an adapter, then the user will need to enter the IP address of the bridge or adapter followed by the backplane port (for example 1) and the slot number of the device. Each item is separated by a comma. As an example; to connect to a Allen Bradley Flex module (via the Flex Adapter at IP address 192.168.1.100) that is in slot 2 of the Flex backplane, the user will need to enter the following path: 192.168.1.100,1,2 (IP address, port (backplane), slot).
RPI	The requested packet interval (RPI) is the rate in milliseconds at which the data will be sent from the originator to the target and vice versa.
Input (T=>O) – Instance	The instance of the input assembly.
Input (T=>O) – Size (bytes)	The size in bytes of the input assembly.
Output (O=>T) – Instance	The instance of the output assembly.
Output (O=>T) – Size (bytes)	The size in bytes of the output assembly.
Configuration – Instance	The instance of the configuration assembly.
Configuration – Size (bytes)	The size in bytes of the configuration assembly. NOTE: This is a read-only value and will be equal to the number of bytes entered into the configuration window below.
Electronic Keying	
Keying	Electronic Keying can be used to ensure that the target device is the correct device type. **Disabled** Keying is not enabled and no key information will be sent in the connection establishment. **Compatible** Keying has been enabled with compatibility enabled. This will allow devices with older firmware to also establish a connection. **Exact**

	Keying has been enabled and the exact device with specific firmware revision will allow the establishment of the connection.
Vendor ID	The Vendor ID of the target device.
Device Type	The Device Type of the target device.
Product Code	The Product Code of the target device.
Major Revision	The Major Revision of the target device.
Minor Revision	The Minor Revision of the target device.
Advanced (Note: Changing these values is not recommended)	
Tick Time	For unconnected messages, this is the time for each tick to calculate the unconnected Time-Out time.
Time-Out Ticks	The number of ticks before the unconnected message is set for timeout.
Time-Out Multiplier	This is the multiplier of the RPI to define the connection timeout time.
Time-Out	The unconnected message timeout time (read-only)
Transport Trigger Direction	The Transport Trigger direction; **Server** or **Client**.
Configuration	
Data	The configuration data that is sent with the forward open connection establishment. The data will need to be entered as a space-delimited, hexadecimal string. For example: 0A 0D 12 EE The configuration size will increase by one each time a byte is added to the configuration.

Table 3.15 – EtherNet/IP Class 1 Connection Parameters

3.10.2.2. IMPORT FROM ONLINE CONTROLLER

Here the EtherNet/IP connection parameters are imported directly from an online Logix controller.

PREPARATION

Before the connection information can be imported, some preparation is required using Studio5000 and a Logix controller:

1. In Studio5000 create a new project and add the required EtherNet/IP device in the IO tree. If the device's profile supports configuration, then configure the device as required.
2. Download the project to a Logix controller.

NOTE: When instantiating modules in Studio5000 do not make use of the "Rack Optimization" communication format.

NOTE: Some versions Logix (V32+) do not support the reading of the module's configuration. Where possible use an earlier version (e.g. V24).

NOTE: It is possible that not all the connection information will be imported as it may not be available due to the type of device and Logix version.

IMPORT CONNECTION PARAMETERS

The connection parameters can be imported from the Logix controller by selecting the ***Import from Online Controller*** option located under the ***Import*** menu of the Class 1 Connection form.

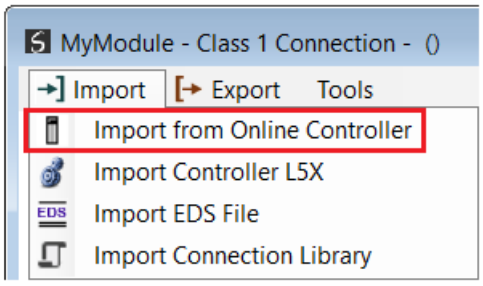


Figure 3.78 – Import from Online Controller

The Import Connection Parameters form will open.

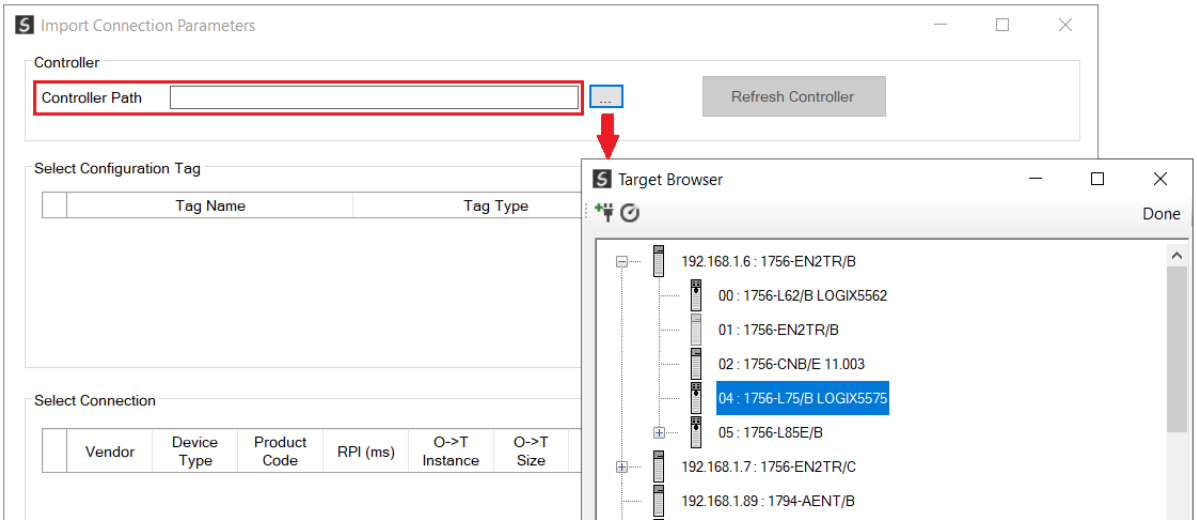


Figure 3.79 – Import Connection Parameters – Controller Path

Enter the path to the Logix controller. This can be either entered manually, or the Browse button "…", can be selected to launch the Target Browser, where the Logix controller can be selected.

Once the Logix controller path has been selected, all the device configuration tags and device connections will be read from the controller and displayed in the Configuration Tag grid and Connection grid respectively.

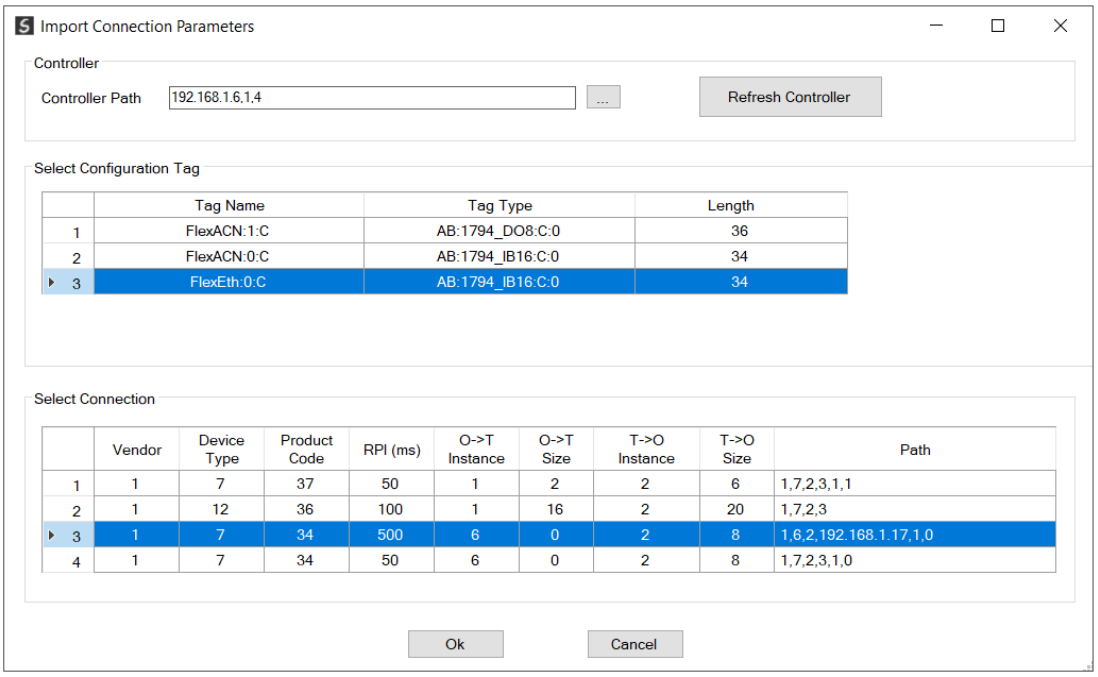


Figure 3.80 – Import Connection Parameters – Select Connection

In order to import all the necessary connection information, the user will need to select both the appropriate ***Configuration Tag***, and the matching ***Connection***.

The new connection's configuration data is derived from the selected ***Configuration Tag***, when the new connection's parameters are derived from the selected ***Connection***.

Once the appropriate selections have been made, press ***Ok***. The imported data will be populated into the Connection form.

The user can then modify the ***Connection Name***, ***Path*** and ***RPI*** as required.

3.10.2.3. IMPORT FROM CONTROLLER L5X FILE

Here the EtherNet/IP connection parameters are imported from a Logix controller's L5X file.

PREPARATION

Before the connection information can be imported some preparation is required using Studio5000:

1. In Studio5000 create a new project and add the required EtherNet/IP device in the IO tree. If the device's profile supports configuration, then configure the device as required.
2. Save the Studio5000 project as an L5X file.

NOTE: When instantiating modules in Studio5000 do not make use of the "Rack Optimization" communication format.

NOTE: It is possible that not all the connection information will be imported as it may not be available in the L5X file due to the type of device and Logix version.

IMPORT L5X FILE

The connection parameters can be imported from the L5X file by selecting the ***Import Controller L5X*** option located under the ***Import*** menu of the Class 1 Connection form.

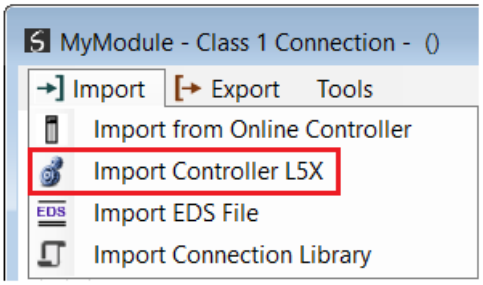


Figure 3.81 – Import from Controller L5X

The Import Connection Parameters form will open.

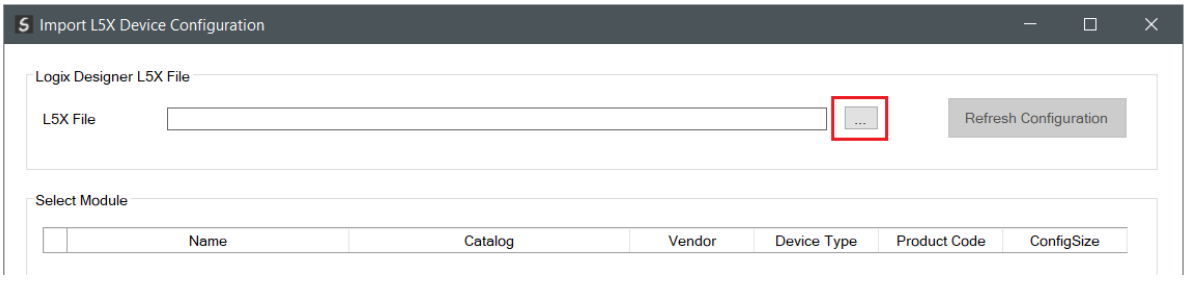


Figure 3.82 – Import L5X Device Configuration – Select L5X

Click on the Browse ("…") button to select the previously generated L5X file.

The modules found in the selected L5X file will then be displayed in the Module List.

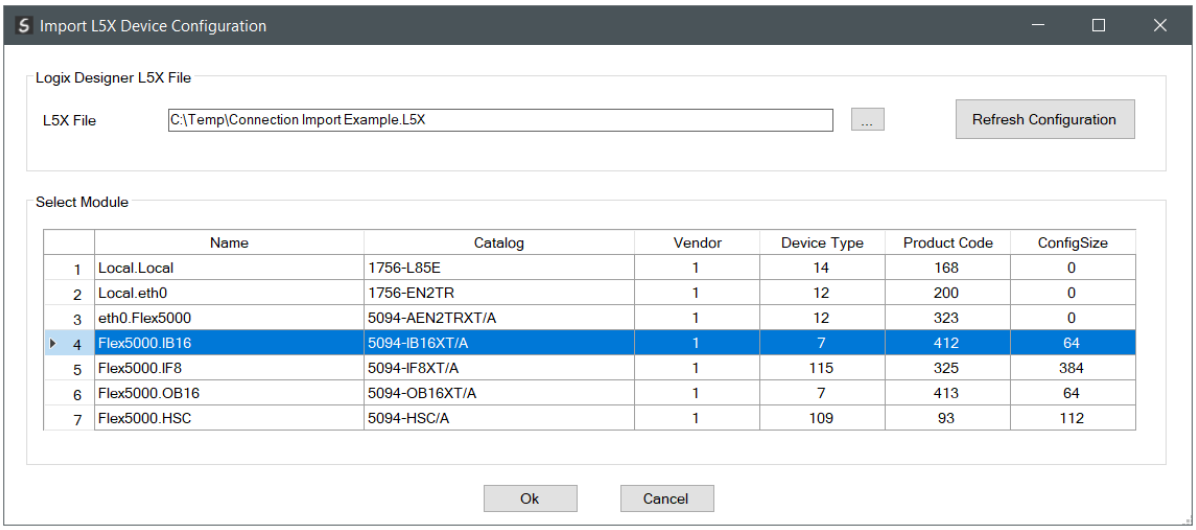


Figure 3.83 – Import L5X Device Configuration

Select the required module and click ***Ok***. The imported data will be populated into the Connection form.

The user can then modify the ***Connection Name***, ***Path*** and ***RPI*** as required.

3.10.2.4. IMPORT EDS FILE

The connection parameters can be imported from a suitable EDS file. Typically, this approach is preferred for devices that do not require configuration data.

To import the connection parameters from a device EDS file, select the ***Import EDS File*** option located under the ***Import*** menu of the Class 1 Connection form.

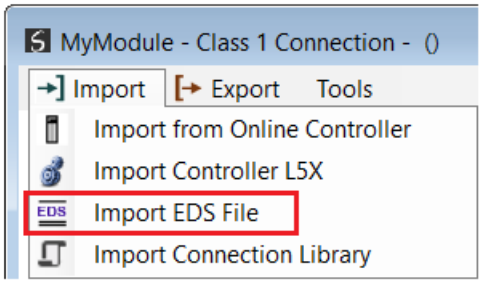


Figure 3.84 – Import EDS File

A File Open dialog will open allowing the user to select the EDS file.

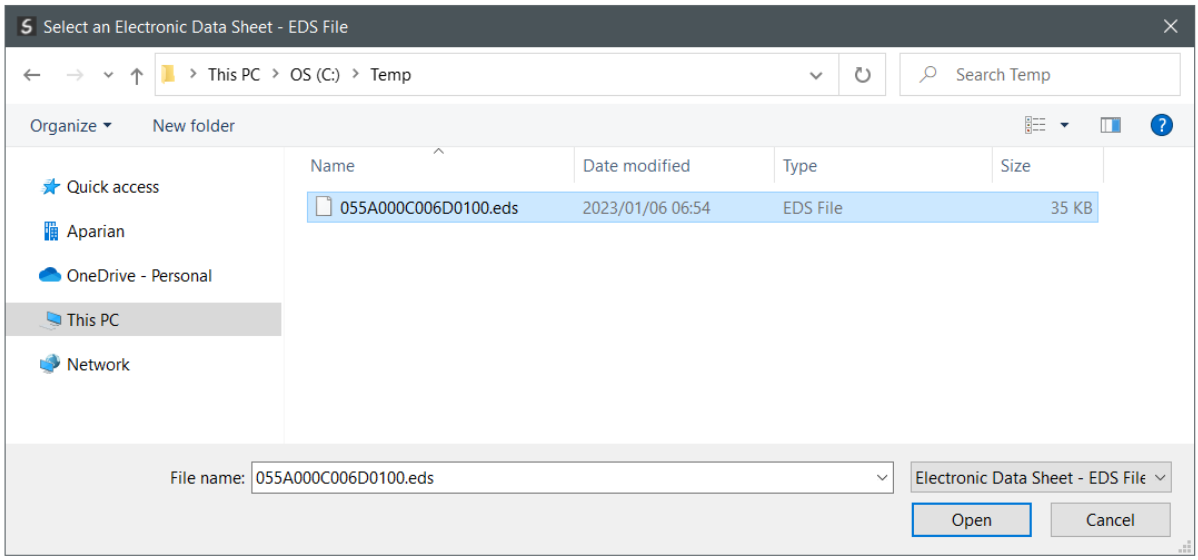


Figure 3.85 – Browse to EDS File

The selected EDS file will be imported, and a summary of the connections displayed. The user will need to select one of the IO connections.

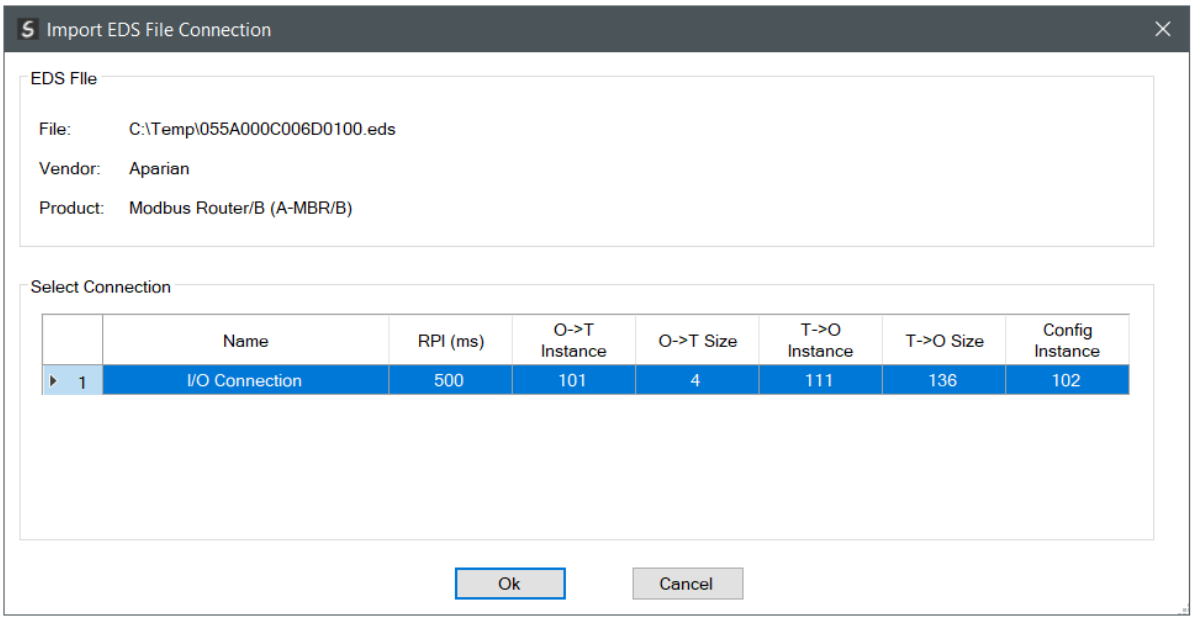


Figure 3.86 – Select Connection

The selected connection within the EDS file will be used to populate the Connection parameters.

The user can then modify the ***Connection Name***, ***Path*** and ***RPI*** as required.

3.10.2.5. IMPORT CONNECTION LIBRARY

The connection parameters can be imported from a previously created Connection Library (.EIPCNX) file.

NOTE: Please contact support to receive a pack of the latest Connection Library files, for commonly used devices.

To import the connection parameters from a Library file, select the ***Import Connection Library File*** option located under the ***Import*** menu of the Class 1 Connection form.

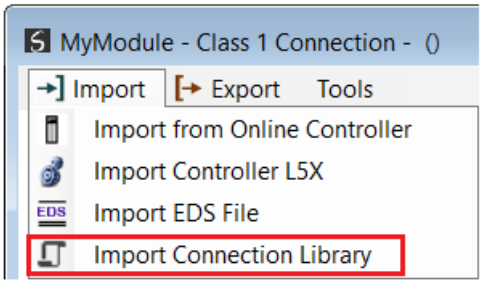


Figure 3.87 – Import Connection Library File

A File Open dialog will open allowing the user to select the Library (.EIPCNX) file.
The selected Library file will be used to populate the Connection parameters.

The user can then modify the ***Connection Name***, ***Path*** and ***RPI*** as required.

EXPORT LIBRARY FILE

In order to create a Library file for future use, select the ***Export Connection Library*** option located under the ***Export*** menu.

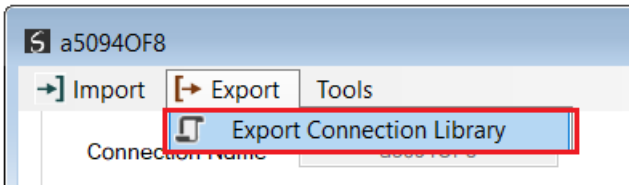


Figure 3.88 – Export Connection Library File

3.11. MODULE DOWNLOAD

Once the CANopen Router/B configuration has been completed, it must be downloaded to the module.

Before downloading the ***Connection Path*** of the module should be set. This path will automatically default to the IP address of the module, as set in the module configuration. It can however be modified, if the CANopen Router is not on a local network.

The Connection path can be set by right-clicking on the module and selecting the ***Connection Path*** option.

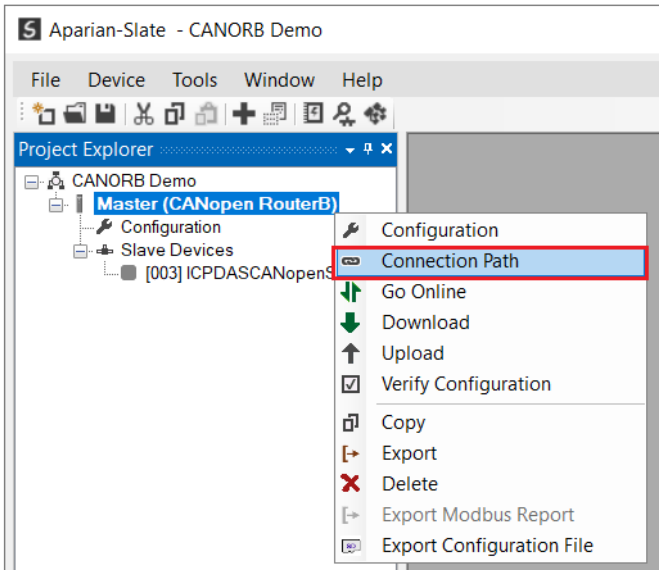


Figure 3.89 - Selecting Connection Path

The new connection path can then be either entered manually or selected by means of the Target Browser.

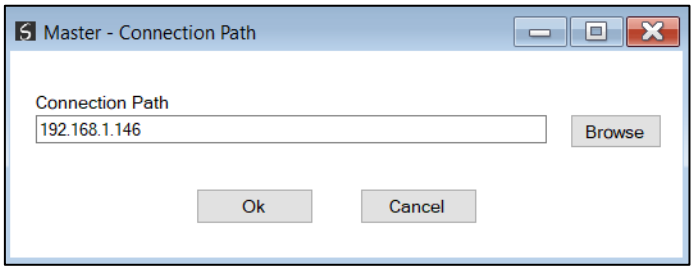


Figure 3.90 - Connection Path

To initiate the download, right-click on the module and select the Download option.

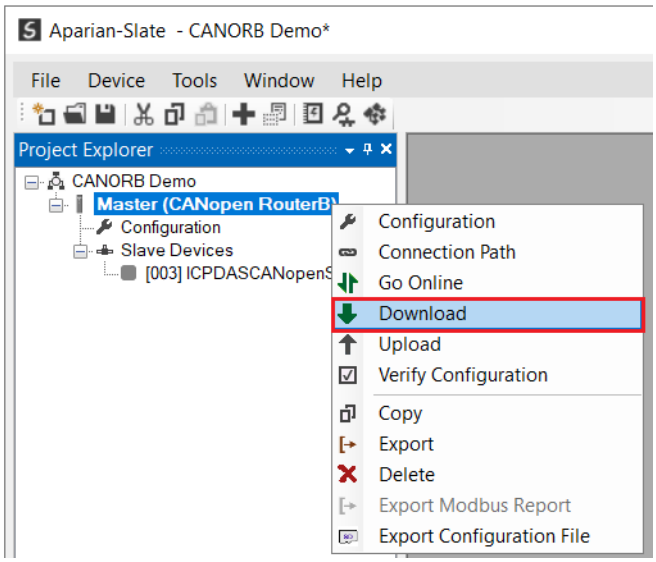


Figure 3.91 - Selecting Download

Once complete, the user will be notified that the download was successful.

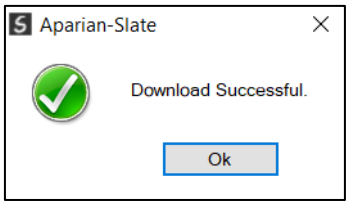


Figure 3.92 - Successful download

Within the Slate environment the module will be in the Online state, indicated by the green circle around the module. The module is now configured and will start operating immediately.

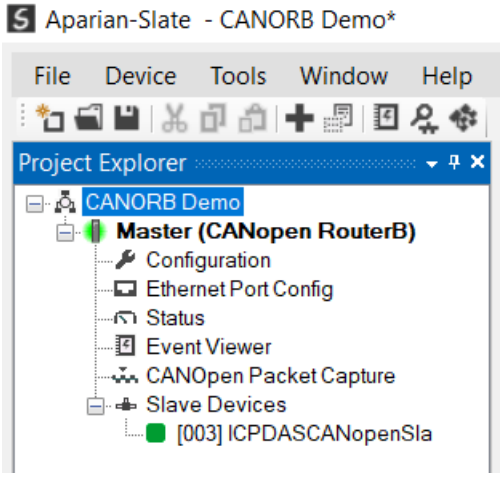


Figure 3.93 - Module online

4. SD CARD

The CANopen Router/B supports an SD Card (see below) which can be used for disaster recovery. The SD Card can be pre-loaded with the required firmware and/or application configuration.

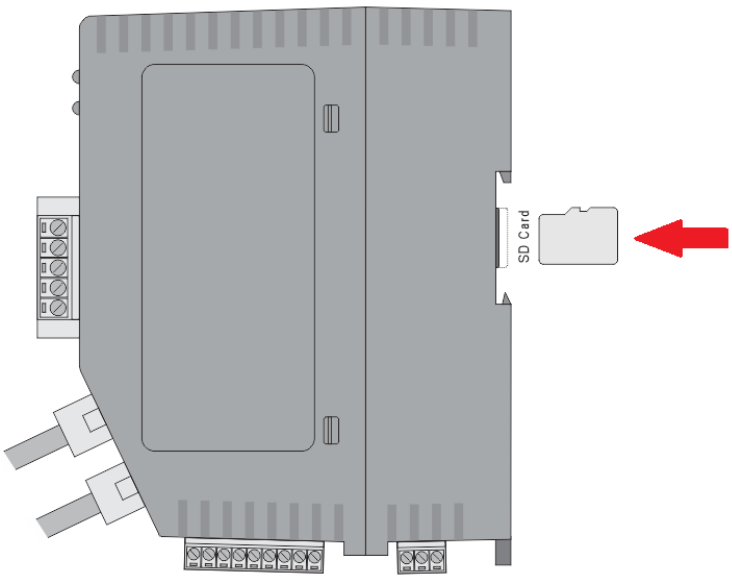


Figure 4.1 – Module Side View – SD Card Slot

NOTE: The user will need to ensure that the SD Card has been formatted for FAT32.

NOTE: All needed files must be copied into the root directory of the SD Card. The module will not use files which are located in folders.

4.1. FIRMWARE

The user can copy the required firmware (which can be downloaded from the Aparian website) onto the root directory of the SD Card.

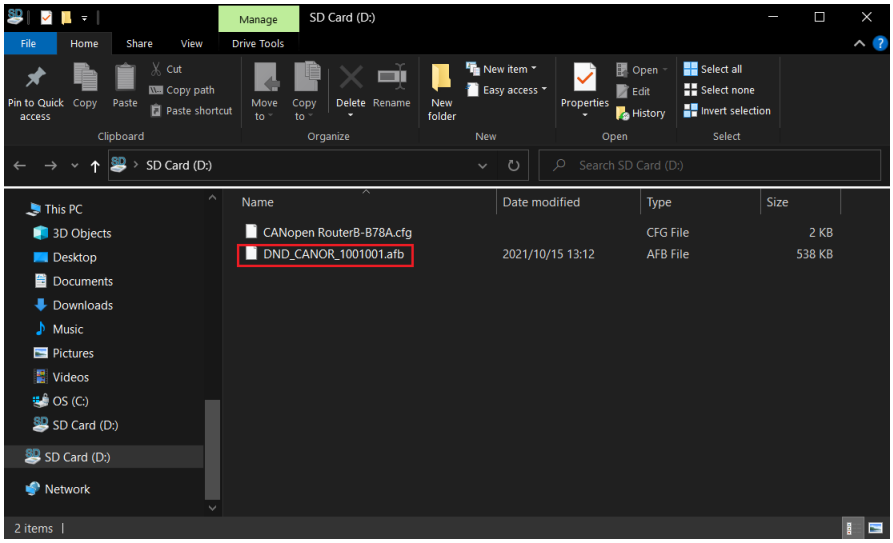


Figure 4.2 – SD Card – Firmware file

NOTE: The filename of the firmware file must not be changed. The specific module will use only the firmware that is valid (e.g. the CANopen Router/B will only use the DND_CANOR firmware file).

NOTE: If more than one firmware file, with different firmware revisions, of the same product is on the SD Card it can cause the module to constantly firmware upgrade the module.

If a faulty module is replaced the user can insert the SD Card with the firmware file on into the new module. While the module is booting it will detect if the firmware on the new module is different from that on the SD Card. If yes, the firmware will either be upgraded or downgraded to the firmware revision on the SD Card.

4.2. NETWORK PARAMETERS

When the SD Card has been inserted into the module and the user is online with the module in Slate, then the user has the option to directly upload the network parameters (e.g., IP Address, Subnet Mask, etc.) on to the SD Card using the *Save Network Parameters to SD Card* option. This will copy the network parameters that has been downloaded to the module directly to the SD Card.

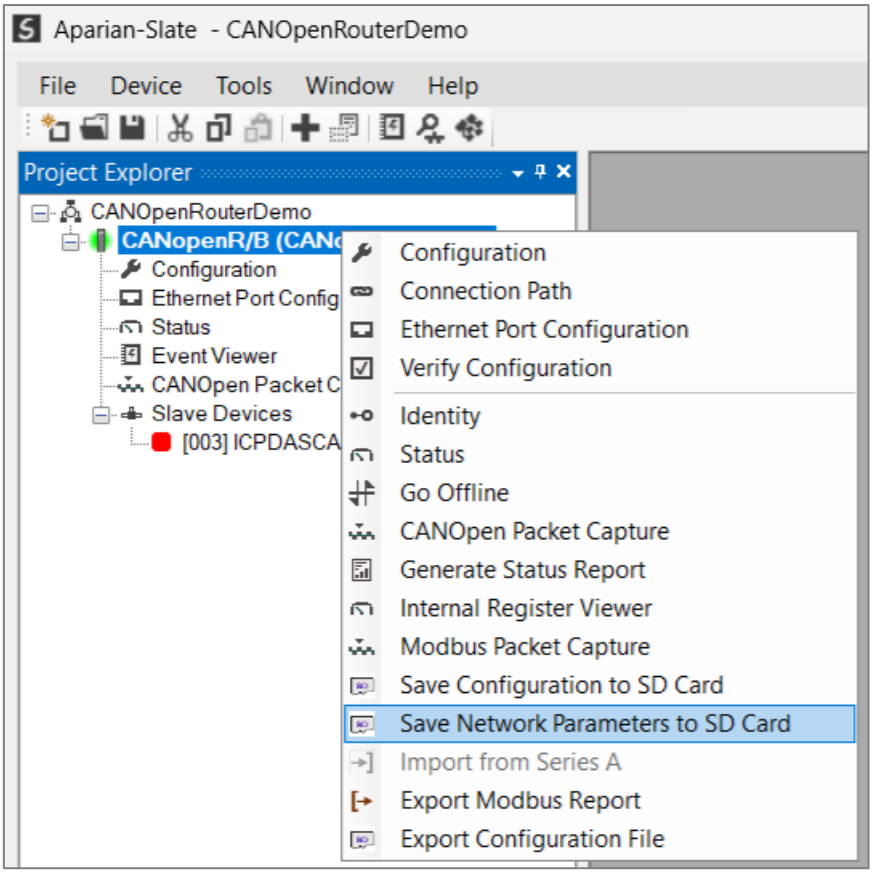


Figure 4.3 – SD Card – Network Parameters

When the module powers up with the SD Card inserted and there are network parameters saved onto the SD Card, it will update the existing network parameters if they are different.

4.3. CONFIGURATION

If a faulty module is replaced the user can insert the SD Card with the configuration file on into the new module. The new module will determine if the configuration on the SD Card is different than the currently loaded configuration (even when there is no configuration on the module). If different, the configuration on the SD Card will be downloaded into the module's NV memory before the module starts executing.

The user can add the Slate configuration file to the SD Card root directory in one of two ways.

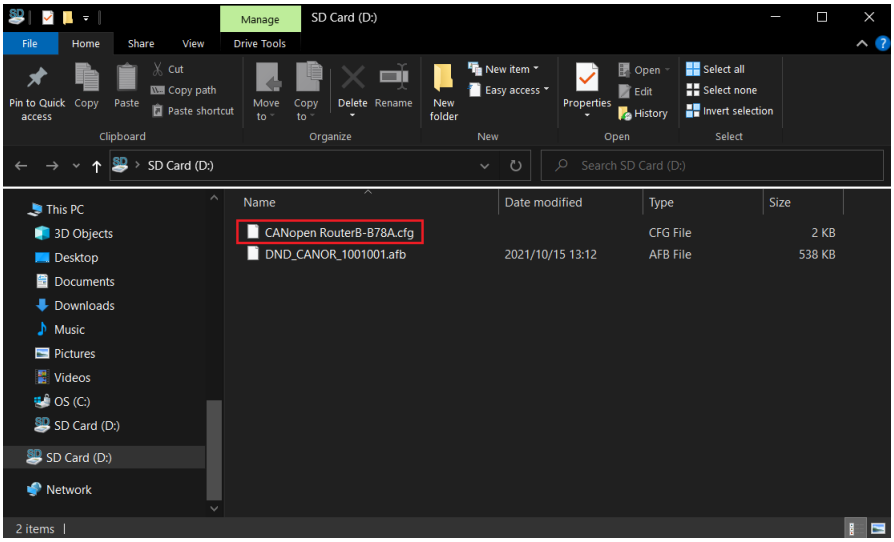


Figure 4.4 – SD Card – Configuration file

4.3.1. Manual Copy

Once the user has created the needed application configuration in the Slate the configuration can be exported to a file that can be used on the SD Card. Once the file has been created the user can copy this file into the root directory of the SD Card.

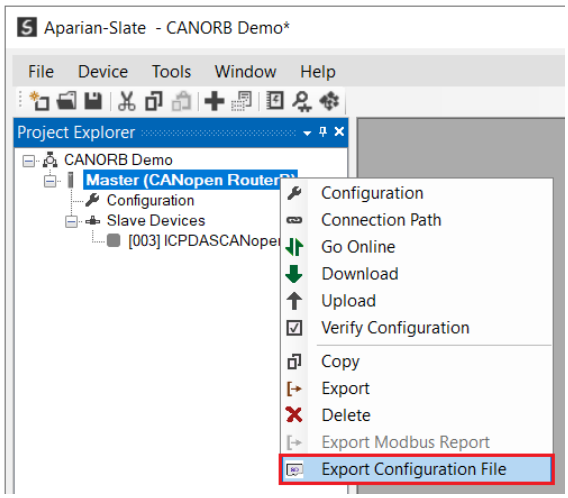


Figure 4.5 – Configuration Export for SD Card

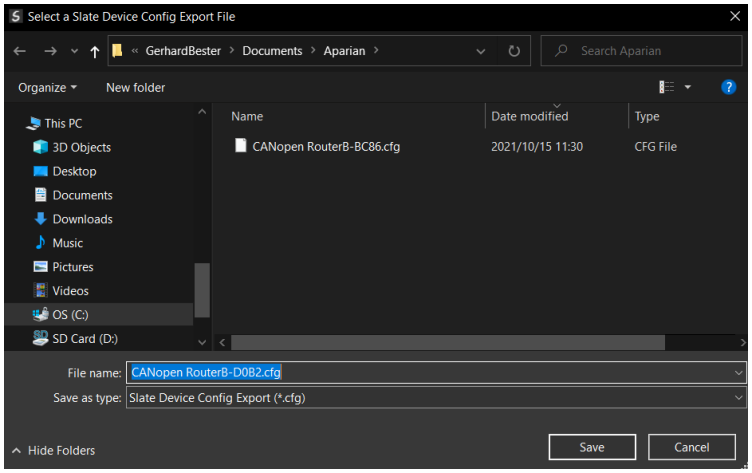


Figure 4.6 – Configuration Export for SD Card

NOTE: The filename of the configuration file must **not** be changed. The specific module will use only the configuration that is valid (e.g. the CANopen Router/B will only use the CANopen configuration file).

NOTE: If more than one configuration file, with different configuration signatures, of the same product is on the SD Card then only the last configuration will be used.

4.3.2. SLATE TRIGGERED UPLOAD

When the SD Card has been inserted into the module and the user is online with the module in Slate, then the user has the option to directly upload the configuration on to the SD Card using the *Save Configuration to SD Card* option. This will copy the configuration that has been downloaded to the module directly to the SD Card without the need to remove it from the module and inserted into a PC.

NOTE: All other configuration files in the SD Card root directory will be deleted when the upload is done.

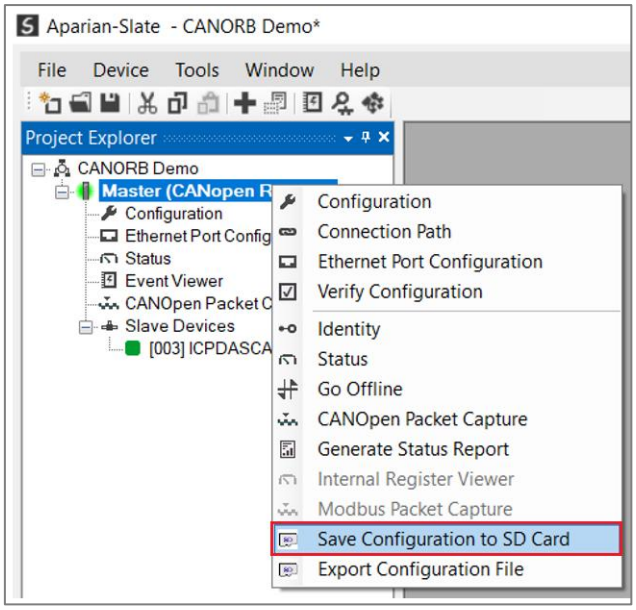


Figure 4.7 – Save Configuration to SD Card

5. DEVICE FIRMWARE UPDATE

The CANopen Router/B module supports in field firmware upgrading. The latest firmware for the module can be downloaded from the Aparian website www.aparian.com. The firmware is digitally signed, so only the correct firmware can be used.

To firmware upgrade the module, follow the steps below:

- From the tools menu in Slate, select the *DeviceFlash* utility.

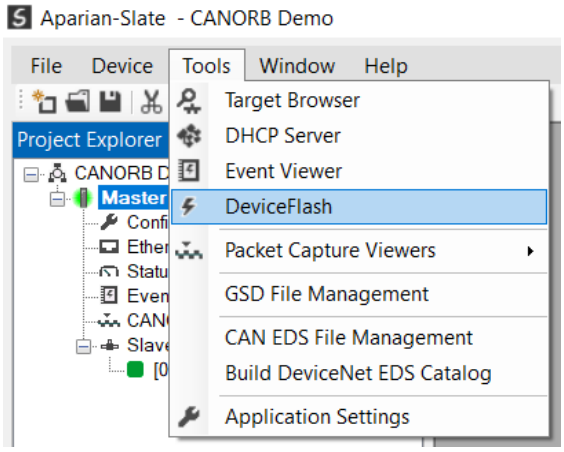


Figure 5.1 – Select DeviceFlash utility from Slate

- When the utility opens, the user will be prompted to select the binary file to be used to firmware upgrade the module.

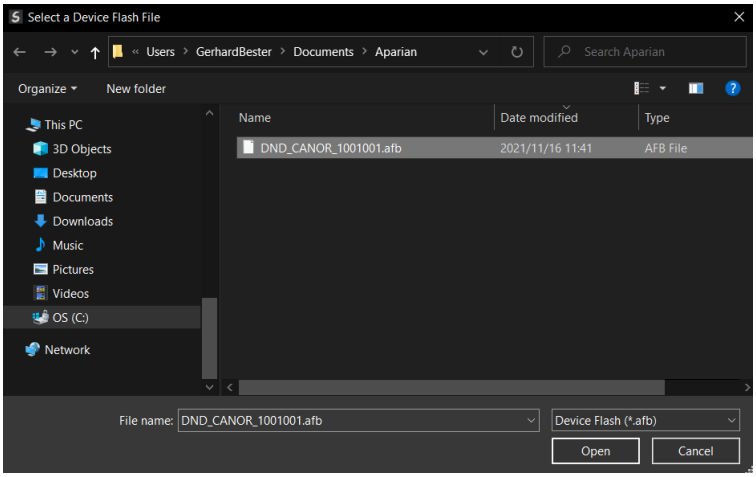


Figure 5.2 – Select the binary file

- After selecting the file, the user will be prompted to select the device to firmware upgrade on the local network.

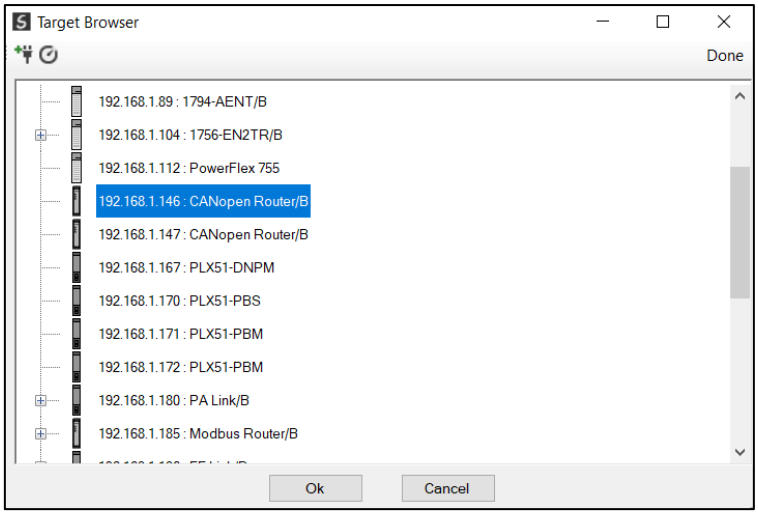


Figure 5.3 – Select the device to be updated

- After the device selection the user will be prompted if the device flash must start. The firmware update will take less than 2 minutes to complete.

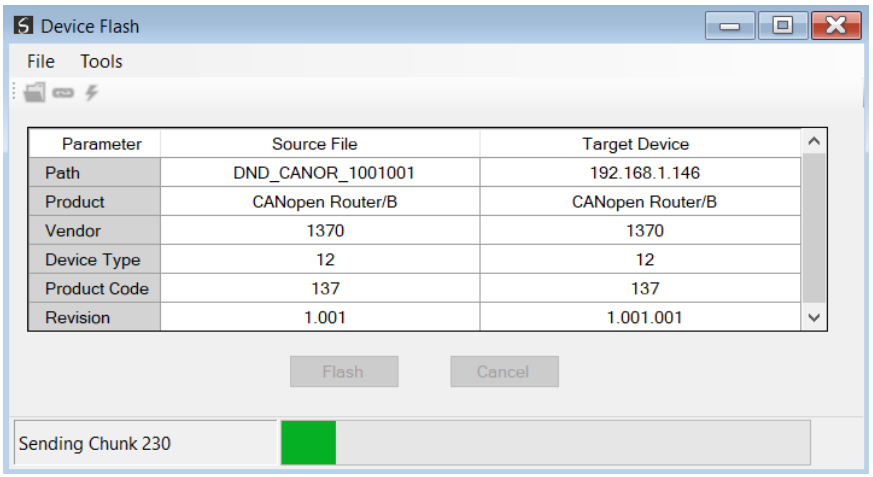


Figure 5.4 – Firmware update busy

- Once the firmware update has successfully completed, the Target Device textboxes will display green.

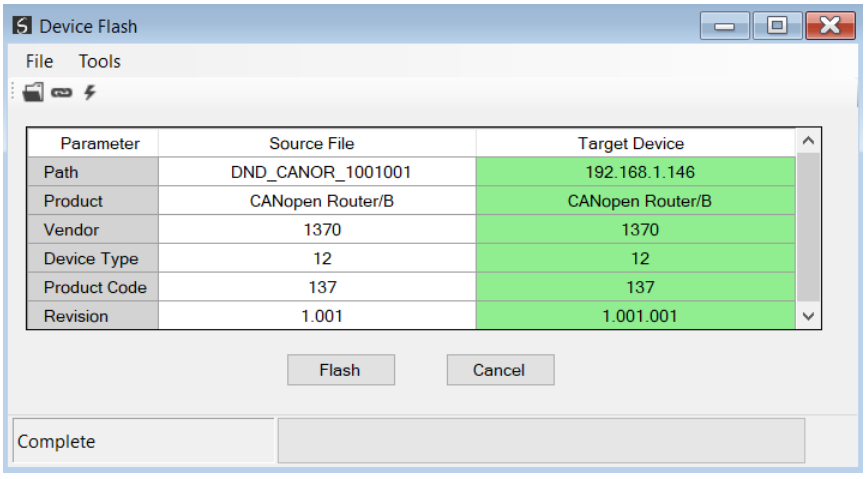


Figure 5.5 – Firmware update successfully completed.

NOTE: If for any reason the firmware update failed (e.g. power down during the update), then the module will revert back to the bootloader. The user can then simply reflash the module again to update it to the latest application firmware.

6. OPERATION

6.1. ETHERNET/IP TARGET

The CANopen Router/B module can operate as a EtherNet/IP target. In this mode the module can exchange CANopen Process Data (PDOs) and Service Data (SDOs) with the connection originator using either Allen Bradley Logix Tag exchange or EtherNet/IP Class 1 data assemblies.

NOTE: Even when using the Logix tag exchange method, the module will still require a Class 1 EtherNet/IP connection from a controller (e.g. Logix controller). When the CANopen Router/B module is a CANopen Master, this is required to set the CANopen Master mode operation (e.g. Operational, Pre-Operational, or Stopped).

6.1.1. ETHERNET/IP CLASS 1 ASSEMBLIES

When the module operates in a Logix "owned" mode the Logix controller will establish a class 1 cyclic communication connection to the CANopen Router. Four input and output assemblies are exchanged at a fix interval (RPI). The UDTs provided will convert the input and output arrays into decorated tag-based assemblies. Refer to the additional information section in this document for the input and output UDTs.

NOTE: The user can use the example Logix mapping routine and UDTs to import the UDTs and mapping routines into a Logix controller.

NOTE: Data being received on CANopen will be written to the **input** assembly data and data that must be sent on the CANopen network will be read from the **output** assembly data.

NOTE: If communication to the controller is lost, then (when in Master mode) the CANopen Router will either force the CANopen network to the pre-operational state, inhibit CANopen communication, or CANopen communication will continue as per normal. The *Interface Comms Fail Mode* parameter in the CAN Bus configuration in Slate with be used to determine the fail state when the communication to the primary interface is lost.

6.1.1.1. INPUT ASSEMBLY 1

The following parameters are used in input assembly 1 of the module.

Parameter	Datatype	Description
InstanceNameLen	DINT	This parameter is the instance name length of the module that was configured under the general CANopen Router/B configuration in Slate.
InstanceName	SINT[16]	This parameter is the instance name of the module that was configured under the general CANopen Router/B configuration in Slate.
Status.ConfigValid	BOOL	Set if a valid configuration is executing in the module.
Status.DuplicateNode	BOOL	Set if a duplicate node is detected on the network.
Status.NetworkOperational	BOOL	The current state of the CANopen network is operational.
Status.NetworkPreOperational	BOOL	The current state of the CANopen network is pre-operational.
Status.NetworkStopped	BOOL	The current state of the CANopen network is stopped.
Status.MasterMode	BOOL	The CANopen Router is operating as a CANopen Master.
Status.SlaveMode	BOOL	The CANopen Router is operating as a CANopen Slave.
Status.MBCommsOnline	BOOL	The Modbus communication (in Modbus Master or Modbus Slave mode) is ok. In Master Master mode, if any of the mapped items in the Modbus Auxiliary Map has failed then this bit will be cleared. In Modbus Slave mode, if Modbus communication has not been received within the Modbus *Inactivity Timeout* then this bit will be cleared.
Status.CANCommsInhibited	BOOL	Module CANopen communication has been inhibited.
Status.BottomPwr	BOOL	Module is receiving power from the bottom power connector.
Status.FrontPwr	BOOL	Module is receiving power from the front CAN connector.
Status.EIPCommsOk	BOOL	The EtherNet/IP communication (in EtherNet/IP Target or EtherNet/IP Originator mode) is ok. In EtherNet/IP Target mode, if any of the Logix tag exchanges has failed or if the Class 1 connection has been lost, then this bit will be cleared. In EtherNet/IP Originator mode, if any of the mapped items in the EtherNet/IP Explicit Message Map has failed or if any of the Class 1 connections are not active, then this bit will be cleared.
TransactionRate	DINT	The transaction rate is the number of CANopen messages per second that the module is currently routing.
DeviceTemperature	REAL	The internal temperature of the CANopen Router module.

UTCTime	DINT[2]	The UTC time on the CANopen network. This has already been formatted for Logix and can be viewed in LINT – Date/Time format.
RxCANCount	DINT	Received CAN message count.
TxCANCount	DINT	Transmitted CAN message count.
CanCrcErrCount	DINT	CAN CRC failed message count.
CanBitErrCount	DINT	CAN Bit error count.
CanStuffErrCount	DINT	CAN Stuff error count.
CanBusOffCount	DINT	The number of times the CAN receiver has detected the Bus Off state.
CanAckErrCount	DINT	The number of times the CAN message was no acknowledged.
CanFormatErrCount	DINT	The number of time a fixed format part of the received frame has the wrong format.
PdoTxCount	DINT	The number of PDO packets transmitted.
PdoRxCount	DINT	The number of PDO packets received.
SdoTxCount	DINT	The number of SDO packets transmitted.
SdoRxCount	DINT	The number of SDO packets received.
TimePcktCount	DINT	The number of TIME packets received or sent.
SyncPcktCount	DINT	The number of SYNC packets received or sent.
EmergencyPcktCount	DINT	The number of EMCY packets received or sent.
HeartbeatPcktCount	DINT	The number of Heartbeat packets received.
TagReads	DINT	The total number of Logix tag reads executed by the module.
TagWrites	DINT	The total number of Logix tag writes executed by the module.
ConnectionFailures	DINT	The number of failed class 3 connection attempts. Note: Logix tag reading and writing requires the module to first establish a class 3 connection with the Logix Controller.
TagErrors	DINT	The number of failed tag access (read/write) requests. These may include privileged violations, non-existing tags, etc.
SlaveMapTransCount	INT[128]	The transaction count for all mapped items in the Virtual Slave Map (when the module is operating as a CANopen Slave). This will increase each time the specific mapped item either transmits or receives the correct PDO.

Table 6.1 - Input assembly 1 parameters

6.1.1.2. INPUT ASSEMBLY 2

The following parameters are used in input assembly 2 of the module.

NOTE: The status information for CANopen Slave Node 1 to 124 will be provided in the Slave Device Mapping. Node 125 – 127 will **not** be shown in the Device Status Registers.

Parameter	Datatype	Description
Slave[**x**] where **x** = SlaveAddress – 1	AparianCANORBNodeInput[124]	The CANopen slaves (node 1-124) are mapped into the Logix input assembly. The below structure will be repeated for each mapped CANopen Slave. NOTE: These fields will only be relevant when the CANopen Router/B is a CANopen Master.
Slave[x].SlaveAddress	SINT	The node address of the mapped slave on the CANopen network.
Slave[x].Reserved	SINT	Reserved
Slave[x].Online	BOOL	When the last response received from the slave is less than the *Slave Inactive Timeout* parameter in the CAN Bus configuration, the slave is considered online, and this bit is set.
Slave[x].ErrRecv	BOOL	Set when the last EMCY message received from the slave has an error.
Slave[x].PDOErr	BOOL	Set if one of the PDOs are not operating correctly.
Slave[x].Initializing	BOOL	Set when the slave is in the initialize state.
Slave[x].Stopped	BOOL	Set when the slave is in the stopped state.
Slave[x].Operational	BOOL	Set when the slave is in the operational state.
Slave[x].PreOperational	BOOL	Set when the slave is in the pre- operational state.
Slave[x].Reserved	BOOL	N/A
Slave[x].PDOMapOk	BOOL	Indication that all the PDOs exchanges are ok.

Table 6.2 - Input assembly 2 parameters

6.1.1.3. INPUT ASSEMBLY 3

The following parameters are used in input assembly 3 of the module.

Parameter	Datatype	Description
CANData	SINT[500]	The first 500 bytes of mapping area in the input assembly where the user can map data from CANopen PDOs and SDOs. In the configuration the user will specify the offset where the data must be in the assembly.

Table 6.3 - Input assembly 3 parameters

6.1.1.4. INPUT ASSEMBLY 4

The following parameters are used in input assembly 4 of the module.

Parameter	Datatype	Description
CANData	SINT[500]	The second 500 bytes of mapping area in the input assembly where the user can map data from CANopen PDOs and SDOs. In the configuration the user will specify the offset where the data must be in the assembly.

Table 6.4 - Input assembly 4 parameters

6.1.1.5. OUTPUT ASSEMBLY 1

The following parameters are used in output assembly 1 of the module.

Parameter	Datatype	Description
GenOperation.NetworkPreOperational	BOOL	When the CANopen Router is the CANopen Master, this bit will force the CANopen network to be PreOperational. **NOTE:** When other NetworkPreOperational and NetworkStop bits are not set then the CANopen Router will set the network state to Operational.
GenOperation.NetworkStop	BOOL	When the CANopen Router is the CANopen Master, this bit will force the CANopen network to be Stopped. **NOTE:** When other NetworkPreOperational and NetworkStop bits are not set then the CANopen Router will set the network state to Operational.
GenOperation.Inhibit	BOOL	Inhibit the CANopen communication.
UTC	DINT[2]	When the CANopen Router is a CANopen Master, the user can write the Logix WallClock time to the UTC tag which will be converted into the CANopen time format for when sending TIME messages.
SlaveModeTPDOOutputTrig	BOOL[128]	When the CANopen Router is operating as a CANopen Slave, these bits are used to trigger sending of TPDOs when the Transmission Type is Evt – Interface. Each time the bit is toggle

		(either from 1 to 0 or from 0 to 1) the respective PDO will send the data to the CANopen Master.

Table 6.5 - Output assembly 1 parameters

6.1.1.6. OUTPUT ASSEMBLY 2

The following parameters are used in output assembly 2 of the module.

Parameter	Datatype	Description
SlaveDevTPDOOutputTrig[**x**] where **x** = SlaveAddress – 1	BOOL[32]	When the CANopen Router is operating as a CANopen Master, these bits are used to trigger sending of RPDOs to the CANopen Slave device (node 1-124) when the Transmission Type is Evt – Interface. Each time the bit is toggle (either from 1 to 0 or from 0 to 1) the respective PDO for the specific device will send the data to the CANopen Slave device.

Table 6.6 - Output assembly 2 parameters

6.1.1.7. OUTPUT ASSEMBLY 3

The following parameters are used in output assembly 3 of the module.

Parameter	Datatype	Description
CANData	SINT[500]	The first 500 bytes of mapping area in the output assembly where the user can map data from CANopen PDOs and SDOs. In the configuration the user will specify the offset where the data must be in the assembly.

Table 6.7 - Output assembly 3 parameters

6.1.1.8. OUTPUT ASSEMBLY 4

The following parameters are used in output assembly 4 of the module.

Parameter	Datatype	Description
CANData	SINT[500]	The second 500 bytes of mapping area in the output assembly where the user can map data from CANopen PDOs and SDOs. In the configuration the user will specify the offset where the data must be in the assembly.

Table 6.8 - Output assembly 4 parameters

6.1.2. LOGIX TAGS

6.1.2.1. CANOPEN MASTER MODE

When a PDO or SDO mapping item has been configured for Logix tags, each time a PDO or SDO is received it will trigger a Logix write of the received CAN data to the configured Logix tag.

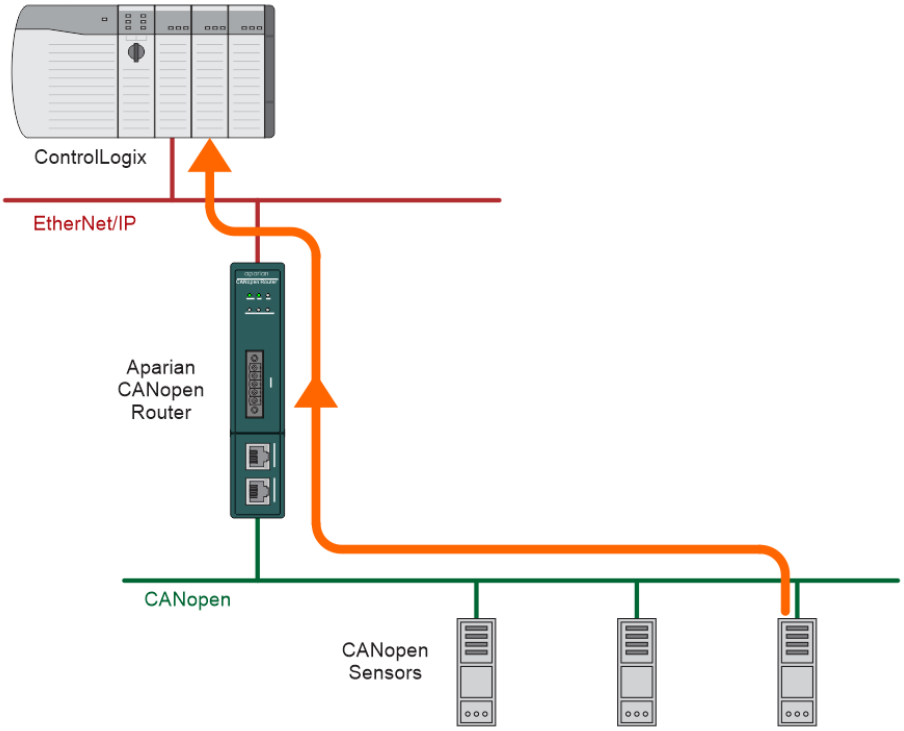


Figure 6.1 – Process variable (TPDO) from slave device to Target Tag

When a PDO or SDO mapping item has been configured for Logix tags, each time a PDO or SDO must be sent (e.g. due to timer interval) it will trigger a Logix read of the configured Logix tag for the CAN data that must be sent.

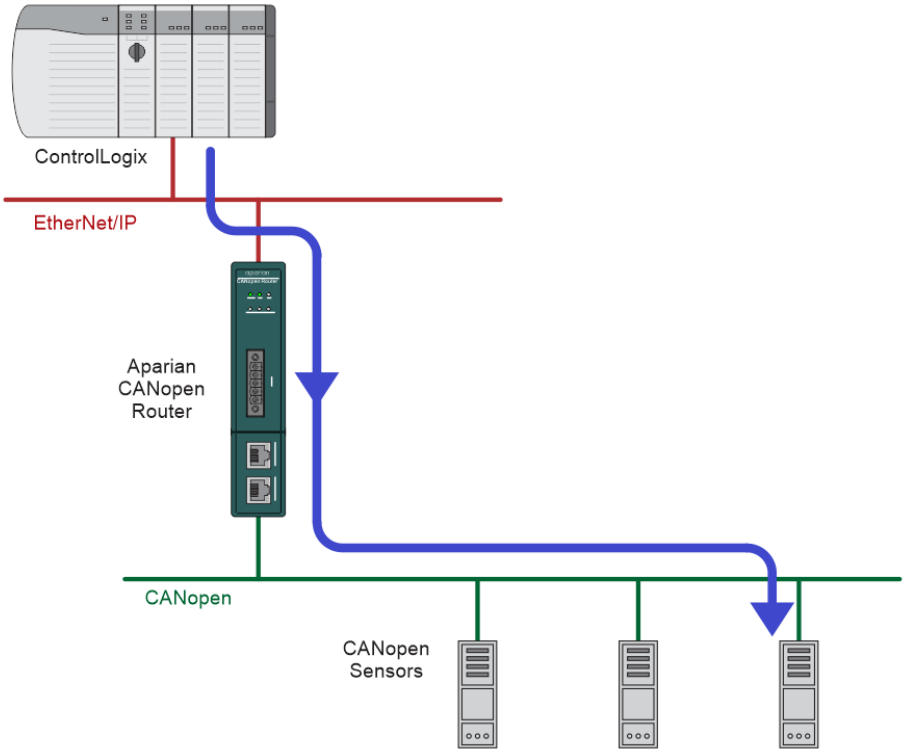


Figure 6.2 – Process variable (RPDO) from Target Tag to slave device

6.1.2.2. CANOPEN SLAVE MODE

When using the EtherNet/IP Target interface with PDO and SDO mapping items configured for Logix tag interface, the PDO data from the CANopen Master will be written into the Target Tag specified in the mapping, and the TPDO data sent to the CANopen Master will be read from the Target Tag specified in the mapping.

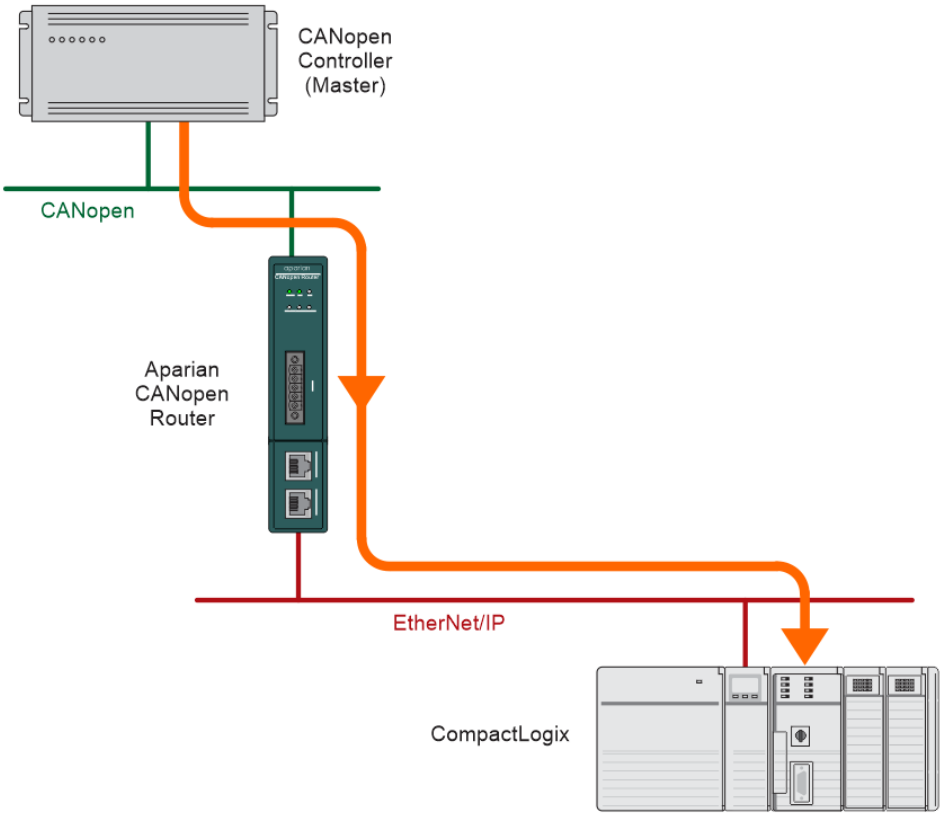


Figure 6.3 – Process variable (RPDO) from CANopen Master to Target Tag

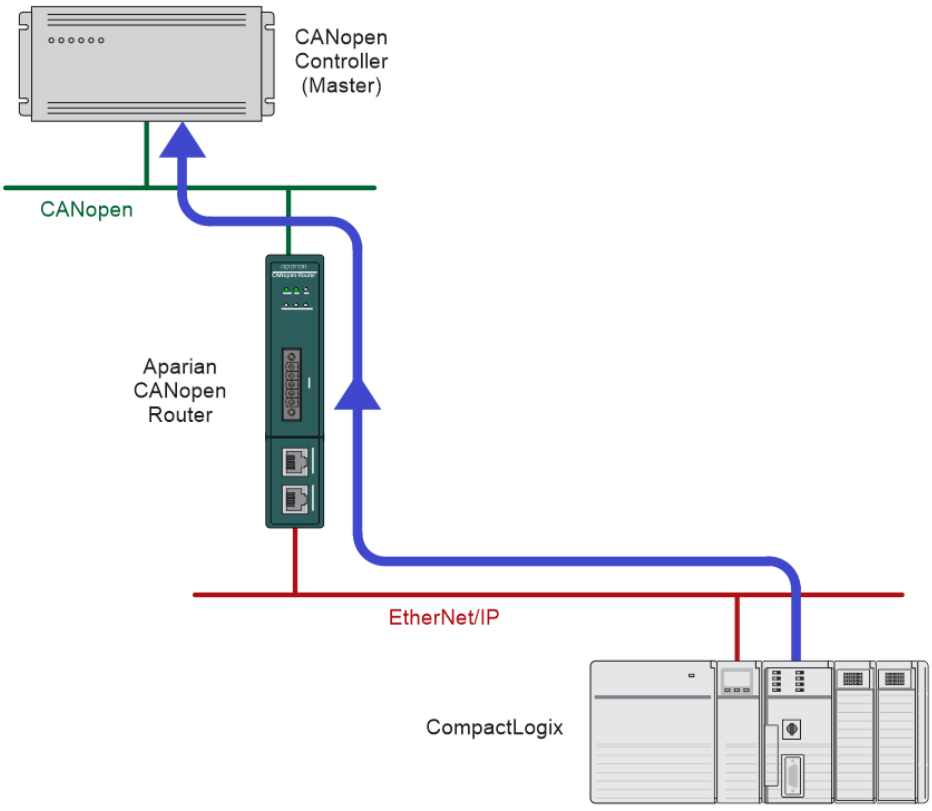


Figure 6.4 – Process variable (TPDO) sent to CANopen Master from Target Tag

NOTE: The user must ensure that the selected Logix tag is sufficiently large to accommodate the specified PDO. For example, if the PDO returns two REAL values, then the Logix Target Tag cannot be only one REAL.

6.1.3. CIP MESSAGING

The CANopen Router/B allows the user to read or write SDO data from and to the CANopen Slave devices using CIP messages. The required parameters for SDO data extraction from a slave device are listed below.

NOTE: Two CIP messages are supported for SDO passthrough messaging. Both have the exact same functionality, the only difference is that the parameters of version 2 (of the SDO passthrough messaging) are 32bit aligned.

6.1.3.1. SDO PASSTHROUGH

A. CIP MESSAGE:

Parameter	Description
Service Code	0x65 (Hex)
Class	0x417 (Hex)
Instance	1
Attribute	N/A
Request Data Length	9 - 489

Table 6.9 – SDO Passthrough Message

B. REQUEST DATA:

Parameter	**Data Type**	**Description**
Node	SINT	The Node Address of the CANopen Slave
Function	SINT	0 – Upload from Slave 1 – Download to Slave
Index	INT	SDO Parameter Index
Sub-index	SINT	SDO Parameter Sub-index
Timeout	INT	The time in milliseconds if not response was received before the request will timeout.

Data Length	INT	The length of the data to follow below (when doing a upload the data length will be zero).
Data	SINT[0-480]	The data to be sent when doing a download.

Table 6.10 – SDO Passthrough Request

C. RESPONSE DATA:

Parameter	Data Type	Description
Status	SINT	This is the status of the request. 0 – Success 1 – Failed 2 – Timeout
Data Length	SINT	The length of the data returned.
Data	SINT[]	The data from the SDO request. The number of bytes will be equal to the Data Length in the response.

Table 6.11 – SDO Passthrough Response

6.1.3.1. SDO PASSTHROUGH (VERSION 2)

A. CIP MESSAGE:

Parameter	Description
Service Code	0x70 (Hex)
Class	0x417 (Hex)
Instance	1
Attribute	N/A
Request Data Length	9 - 489

Table 6.12 – SDO Passthrough V2 Message

B. REQUEST DATA:

Parameter	Data Type	Description
Node	SINT	The Node Address of the CANopen Slave

Function	SINT	0 – Upload from Slave 1 – Download to Slave
Index	INT	SDO Parameter Index
Sub-index	INT	SDO Parameter Sub-index
Reserved	INT	-
Timeout	INT	The time in milliseconds if not response was received before the request will timeout.
Data Length	INT	The length of the data to follow below (when doing a upload the data length will be zero).
Data	SINT[0-480]	The data to be sent when doing a download.

Table 6.13 – SDO Passthrough V2 Request

C. RESPONSE DATA:

Parameter	Data Type	Description
Status	SINT	This is the status of the request. 0 – Success 1 – Failed 2 – Timeout
Data Length	SINT	The length of the data returned.
Data	SINT[]	The data from the SDO request. The number of bytes will be equal to the Data Length in the response.

Table 6.14 – SDO Passthrough V2 Response

6.2. ETHERNET/IP ORIGINATOR

The CANopen Router/B module can operate as an EtherNet/IP originator. In this mode the module can exchange CANopen Process Data (PDOs) and Service Data (SDOs) with EtherNet/IP devices using either the input and output assemblies of the Class 1 EtherNet/IP connection to the device or using an explicit (Class 3 or UCMM) EtherNet/IP message to read or write data.

6.2.1. ETHERNET/IP CLASS 1 CONNECTIONS

Once the EtherNet/IP Class 1 connections are setup and established then the received CANopen data will be written to the output assembly of selected EtherNet/IP device

(Originator to Target). In the configuration the user will specify the offset where the data must be in the assembly.

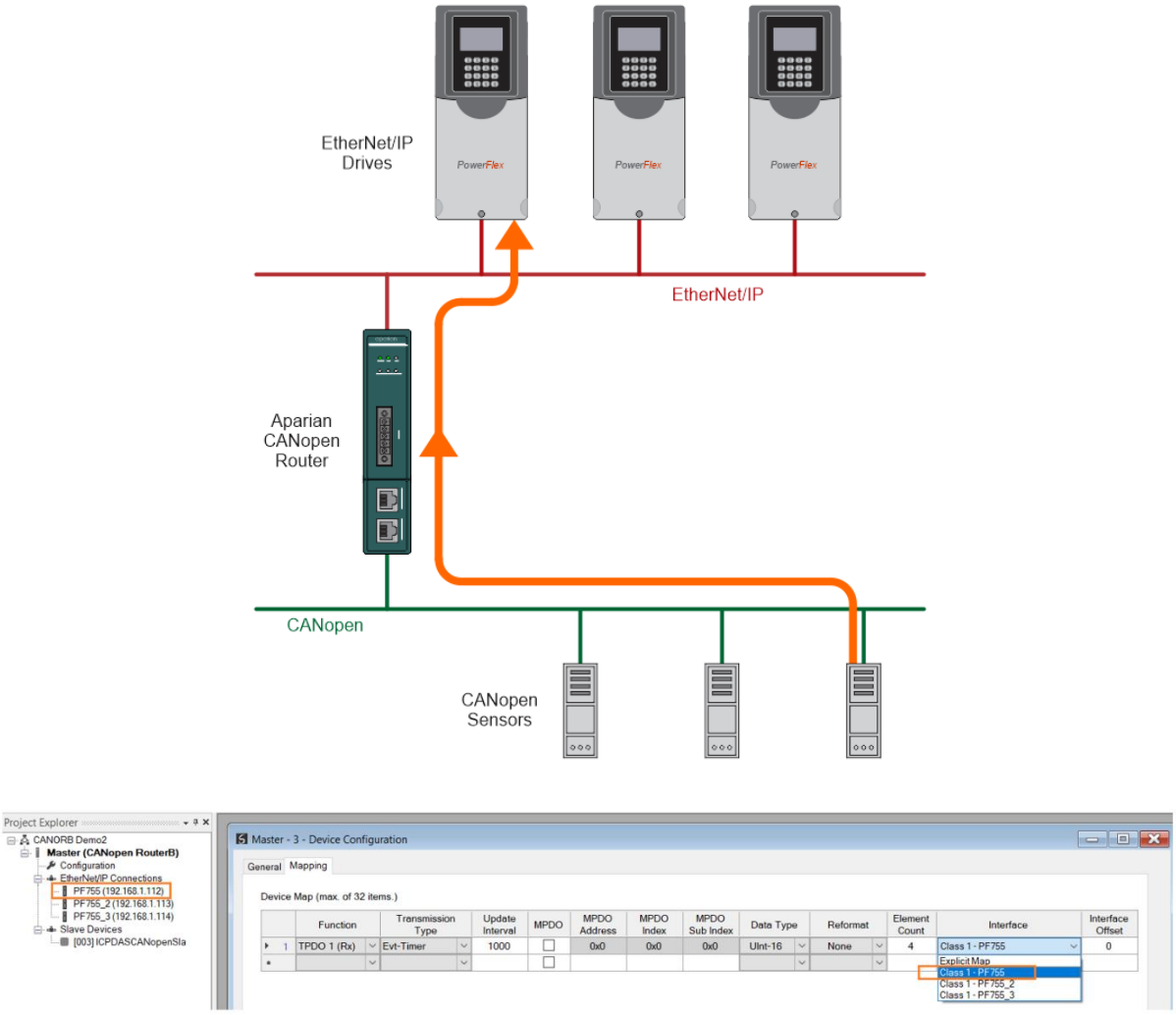


Figure 6.5 – Process variable (TPDO) from CANopen Slave to EtherNet/IP Device

The transmitted CANopen data will be read from the input assembly of the selected EtherNet/IP device (Target to Originator). In the configuration the user will specify the offset where the data must be in the assembly.

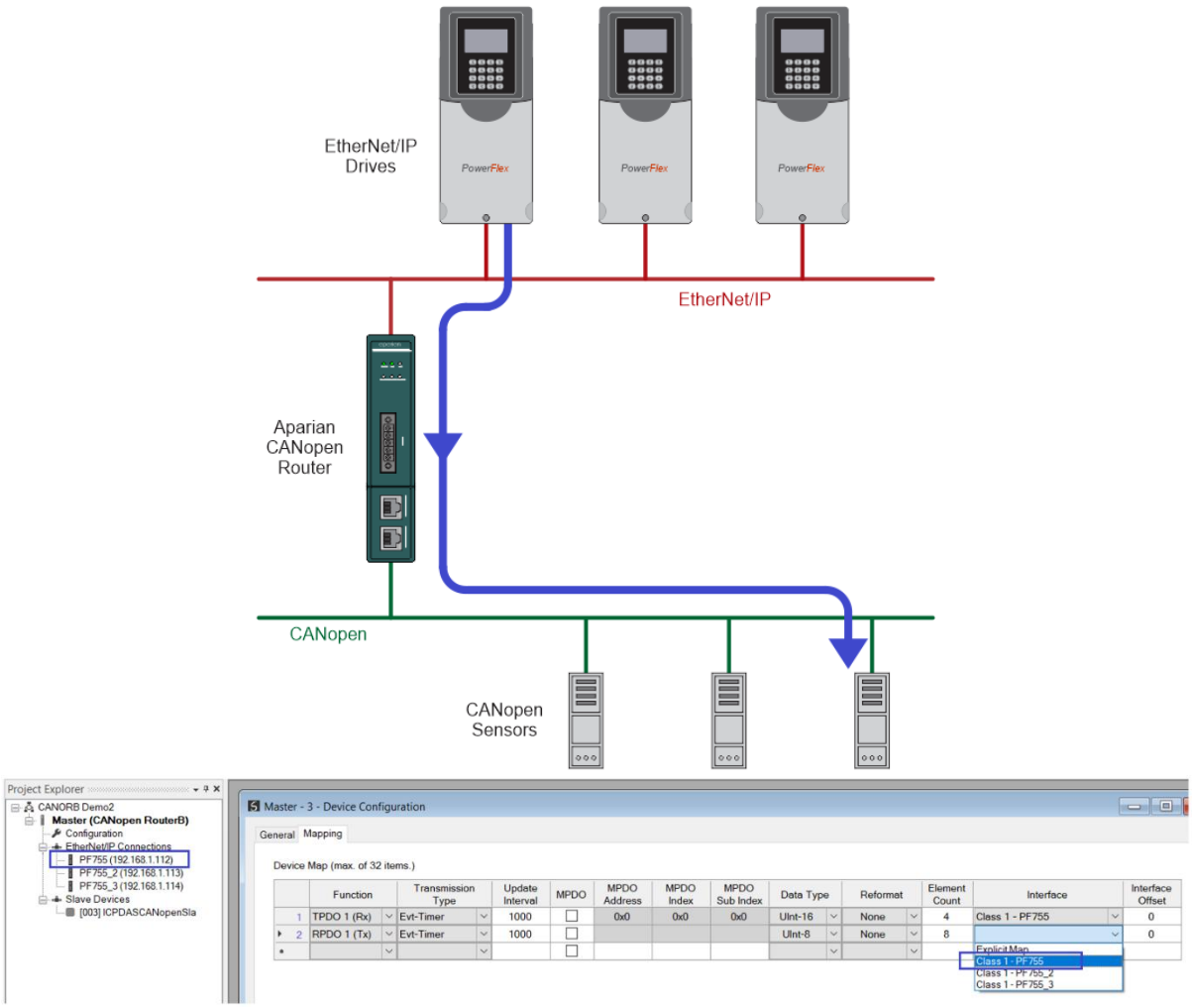


Figure 6.6 – Process variable (RPDO) from EtherNet/IP Device to CANopen Slave

6.2.2. EXPLICIT MESSAGING

When using the EtherNet/IP Explicit Messaging, the user can configure up to five EtherNet/IP devices which will be used for the Explicit Messaging. This is configuration is located in the *EtherNet/IP Devices* tab. Following this, the EtherNet/IP Map of explicit messages needs to be configured. The Explicit Messaging has 100KB internal EIP mapping table where data can be stored for exchanges between the explicit EtherNet/IP devices and the CANopen PDO and SDO data.

When receiving CANopen data (either PDO or SDO) the data is stored at a configured offset in the EIP mapping table. When sending CANopen Data (either PDO or SDO) the data is retrieved from the EIP mapping table at the configured offset. The offset appears as the ***Interface Offset*** in either the SDO or PDO mapping.

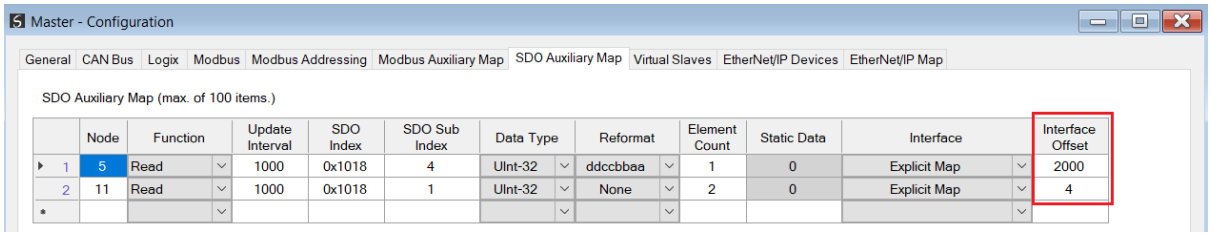


Figure 6.7 – EIP Table Interface Offset for SDO

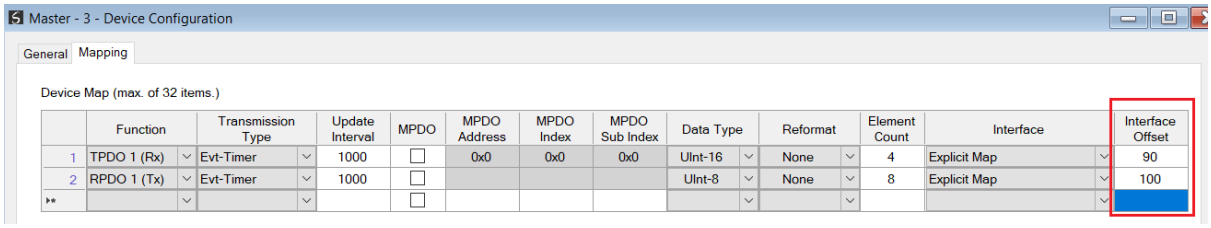


Figure 6.8 – EIP Table Interface Offset for PDO

When reading data from the EtherNet/IP device (for example using a Get function in the EtherNet/IP Mapping) the data is stored at a configured offset in the EIP mapping table. The offset is configured by configuring the interface ***Input Offset*** in the EtherNet/IP Map.

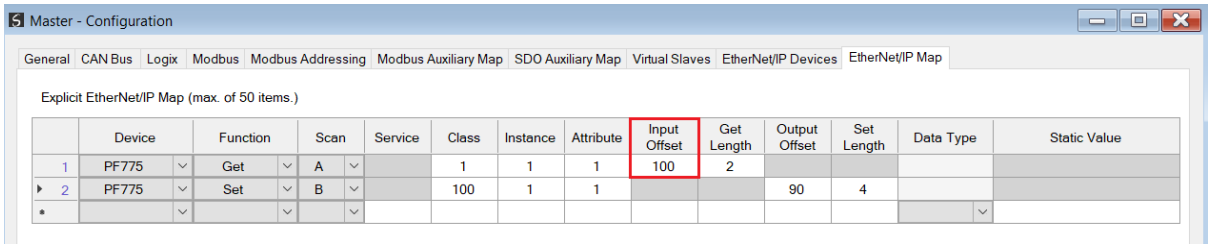


Figure 6.9 – EIP Table Interface Input Offset for EtherNet/IP Map

When sending data to an EtherNet/IP device (for example using a Set function in the EtherNet/IP Mapping) the data is retrieved from the EIP mapping table at the configured offset. This offset appears as the interface ***Output Offset*** in the EtherNet/IP Map.

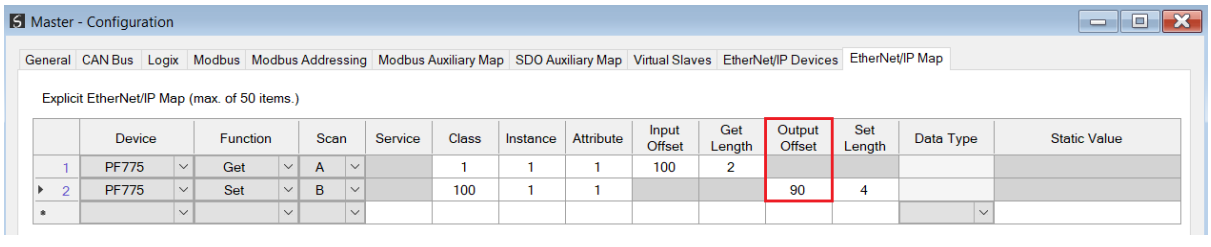


Figure 6.10 – EIP Table Interface Output Offset for EtherNet/IP Map

When sending and receiving data for a EtherNet/IP device (for example using a Custom function in the EtherNet/IP mapping) then both interface ***Input Offset*** and ***Output Offset*** will need to be configured for the data from the EIP mapping table.

Below is an example of mapping a EtherNet/IP device (PF775) Get and Set functions to CANopen RPDO (Rx) and CANopen TPDO (Rx).

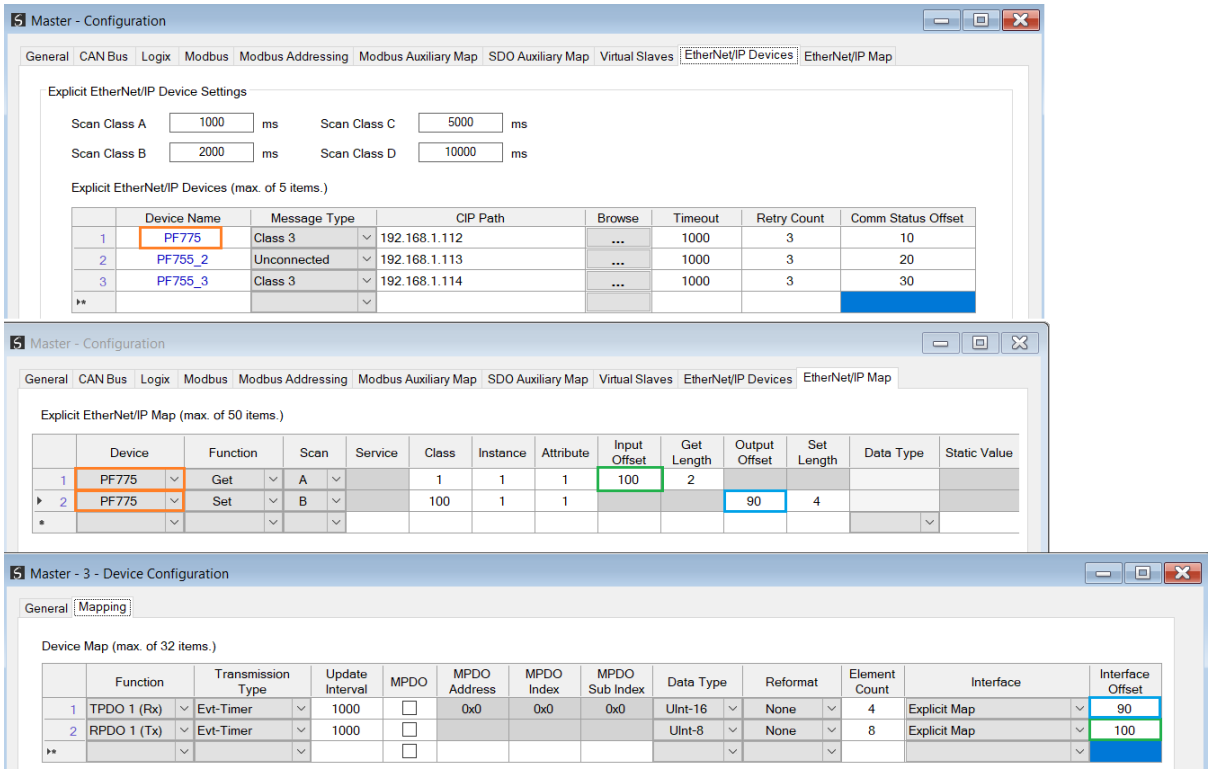


Figure 6.11 – EtherNet/IP Explicit Message Operation

6.3. MODBUS MASTER

6.3.1. OPERATION

When the CANopen Router/B module's primary interface is set to Modbus Master, then the CANopen data (PDO or SDO) can be mapped to and from configurable internal Modbus Registers and offsets.

The internal Modbus Registers are then asynchronously exchanged with up to 20 Modbus devices as configured in the Modbus Auxiliary Map. In this mapping the user can read or write data from the internal Modbus Registers to a remote Modbus device.

In the example below the CANopen Router (as a CANopen Master) with the primary interface set to Modbus Master will read multiple Modbus Holding Registers from a Modbus Slave device and then transmit that data to a CANopen Slave device.

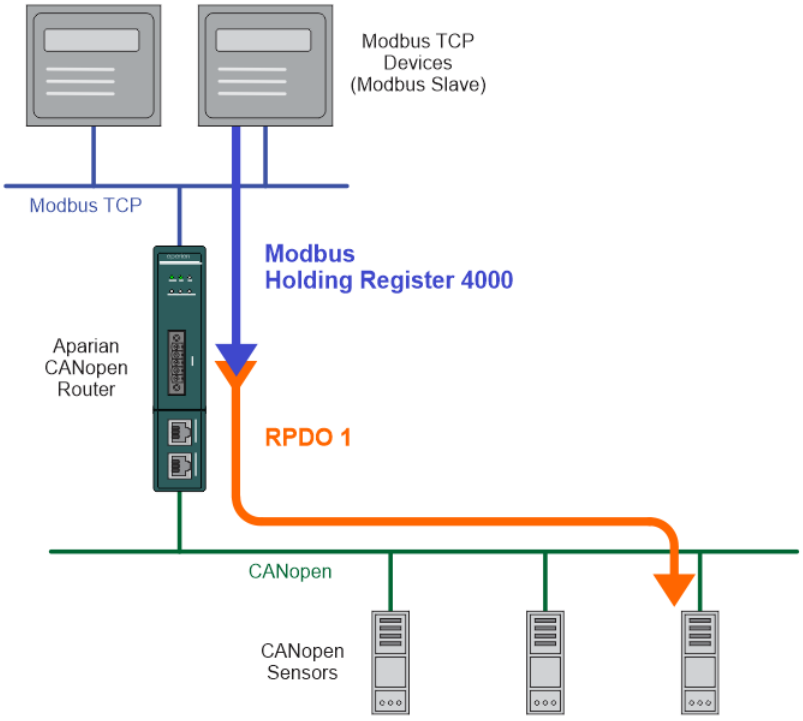


Figure 6.12 – Modbus Master to CANopen Slave operation

For this example the user will configure the Modbus Auxiliary Map to read data from a Modbus Slave device. The CANopen Router/B will request data from Modbus Holding Register 4000 (from the external Modbus Slave) and write it to the module's internal Modbus Holding Register 3000.

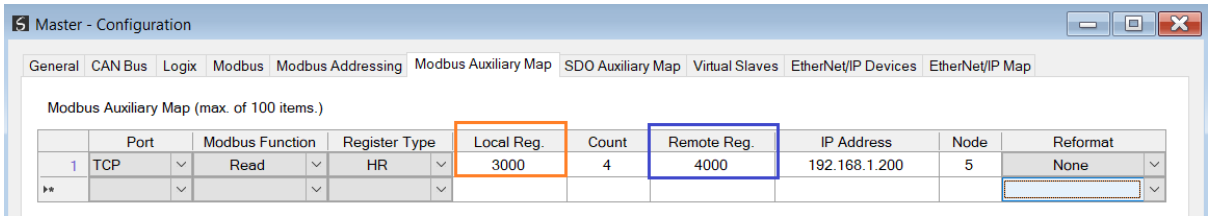


Figure 6.13 – Modbus Master Aux Mapping

Next the RPDO (Tx) for the slave device needs to be mapped to the internal Modbus Holding Register (offset 3000) which is where the data received from the Modbus Slave was stored.

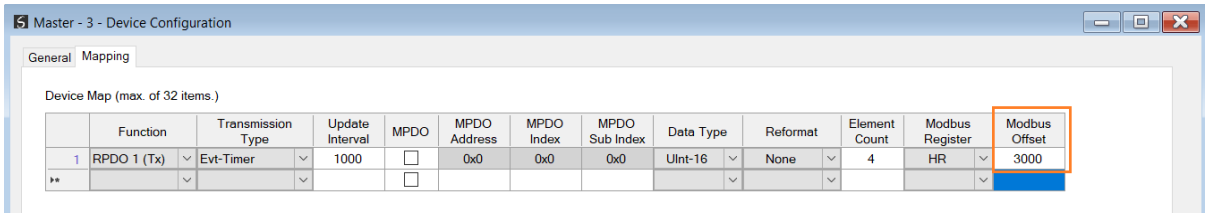


Figure 6.14 – CANopen Slave Modbus Mapping

NOTE: Every PDO can consume four Modbus Holding Registers, because the max PDO size is 8 bytes which equates to 4 Modbus words.

NOTE: The user will need to ensure that when writing to the CANopen Router Modbus Holding Registers that the registers holding data from the device are not inadvertently overwritten.

6.3.2. Fixed Modbus Mapping

When the CANopen Router/B is operating as a Modbus Master it will provide status and control via configurable Modbus Registers.

The Master Status and Slave Device can be written to either Modbus Holding Register or Modbus Input Register at a configured offset. The Status data will be populated by the CANopen Router/B module.

The Control and TPDO Trigger data will be read from a the configured Modbus Holding Registers. The Control and Trigger data will need to be populated by the external Modbus device.

NOTE: When the CANopen Router/B is a CANopen Master then the CANopen network state and operation can be controlled in the Master Control Registers.

NOTE: When the CANopen Router/B is a CANopen Slave and a TPDO transmission type in the Virtual Slave map is set to *Evt-Interface* then the

emulated TDPOs in the Virtual Slave map can be triggered for transmission by toggling the bit values in the *SlaveModeTPDOOutputTrig* field.

NOTE: When the CANopen Router/B is a CANopen Master and the CANopen Slave device RPDO (Tx) transmission type has been set to *Evt-Interface*, then the respective PDO can be triggered for transmission by toggling the bit values in the TPDO trigger Registers.

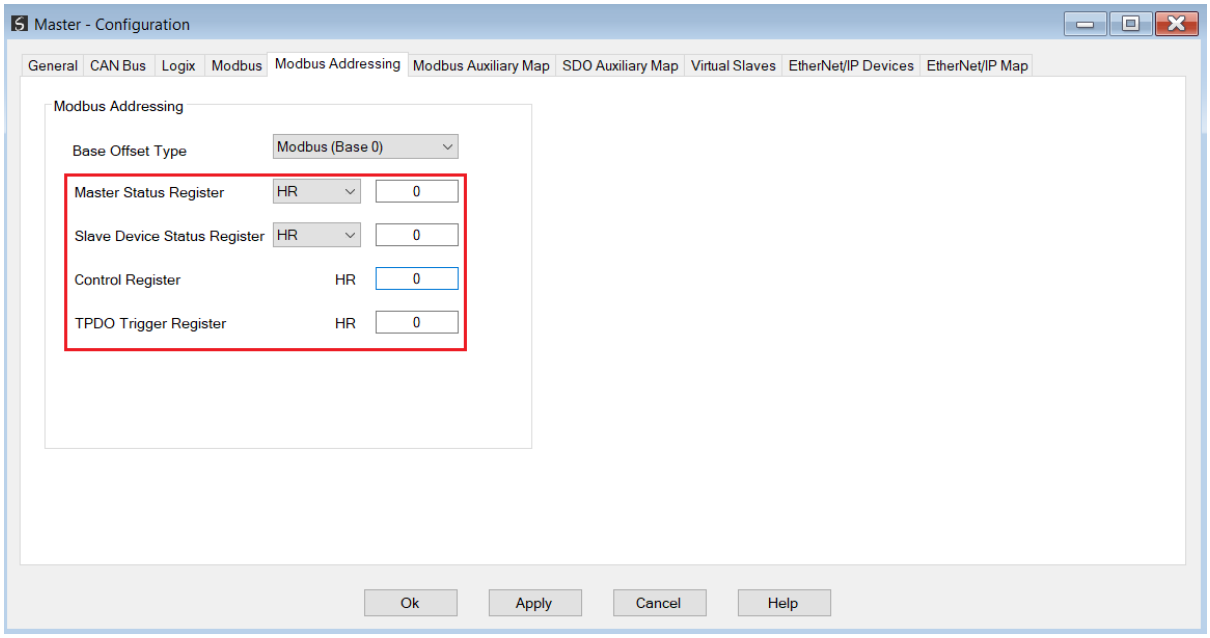


Figure 6.15 – Modbus Status and Control Addressing

A summary of the utilized Modbus Registers can be viewed by right-clicking on the module and selecting the ***Export Modbus Report*** option. This produces a CSV (comma-separated-variable) file showing the Modbus registers used.

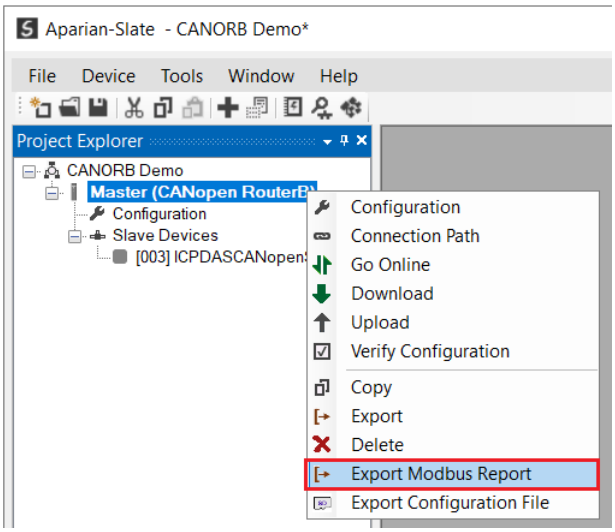


Figure 6.16 – Export Modbus Report

Below is the format of each of the fixed Modbus Registers.

6.3.2.1. MASTER STATUS REGISTER

The Master Status register will start at the configured offset (in the *Modbus Addressing* configuration) and each parameter will then have a HR/IR Offset relative to the starting address.

Field	HR/IR Offset	HR/IR Size (16bit words)	Data Type	Description
InstanceNameLen	0	2	DINT	The length of the instance name of the module that was configured under the general CANopen Router/B configuration in Slate.
InstanceName	2	8	SINT[16]	The instance name of the module that was configured under the general CANopen Router/B configuration in Slate.
Status	**10**	**2**	**DINT**	**Status Flags in Boolean Format**
***Status**.ConfigValid*	*10.0*	-	*BOOL*	*Set if a valid configuration is executing in the module.*
***Status**.DuplicateNode*	*10.1*	-	*BOOL*	*Set if a duplicate node is detected on the network.*
***Status**.NetworkOperational*	*10.2*	-	*BOOL*	*The current state of the CANopen network is operational.*
***Status**.NetworkPreOperational*	*10.3*	-	*BOOL*	*The current state of the CANopen network is pre-operational.*
***Status**.NetworkStopped*	*10.4*	-	*BOOL*	*The current state of the CANopen network is stopped.*
***Status**.MasterMode*	*10.5*	-	*BOOL*	*The CANopen Router is operating as a CANopen Master.*
***Status**.SlaveMode*	*10.6*	-	*BOOL*	*The CANopen Router is operating as a CANopen Slave.*
***Status**.MBCommsOnline*	*10.7*	-	*BOOL*	*The Modbus communication (in Modbus Master or Modbus Slave mode) is ok.* *In Master Master mode, if any of the mapped items in the Modbus Auxiliary Map has failed then this bit will be cleared.* *In Modbus Slave mode, if Modbus communication has not been received within*

				the Modbus Inactivity Timeout then this bit will be cleared.
***Status**.CANCommsInhibited*	*10.8*	-	*BOOL*	*Module CANopen communication has been inhibited.*
***Status**.BottomPwr*	*10.9*	-	*BOOL*	*Module is receiving power from the bottom power connector.*
***Status**.FrontPwr*	*10.10*	-	*BOOL*	*Module is receiving power from the front CAN connector.*
***Status**.EIPCommsOk*	*10.11*	-	*BOOL*	*The EtherNet/IP communication (in EtherNet/IP Target or EtherNet/IP Originator mode) is ok.* *In EtherNet/IP Target mode, if any of the Logix tag exchanges has failed or if the Class 1 connection has been lost, then this bit will be cleared.* *In EtherNet/IP Originator mode, if any of the mapped items in the EtherNet/IP Explicit Message Map has failed or if any of the Class 1 connections are not active, then this bit will be cleared.*
***Status**.Reserved*	*10.12-10.31*	-	*BOOL*	*N/A*
TransactionRate	12	2	DINT	The transaction rate is the number of CANopen messages per second that the module is currently routing.
DeviceTemperature	14	2	REAL	The internal temperature of the CANopen Router module.
UTCTime	16	4	DINT[2]	The UTC time on the CANopen network. This has already been formatted for Logix and can be viewed in LINT – Date/Time format.
RxCANCount	20	2	DINT	Received CAN message count.
TxCANCount	22	2	DINT	Transmitted CAN message count.
CanCrcErrCount	24	2	DINT	CAN CRC failed message count.
CanBitErrCount	26	2	DINT	CAN Bit error count.
CanStuffErrCount	28	2	DINT	CAN Stuff error count.
CanBusOffCount	30	2	DINT	The number of times the CAN receiver has detected the Bus Off state.
CanAckErrCount	32	2	DINT	The number of times the CAN message was no acknowledged.
CanFormatErrCount	34	2	DINT	The number of time a fixed format part of the received frame has the wrong format.

PdoTxCount	36	2	DINT	The number of PDO packets transmitted.
PdoRxCount	38	2	DINT	The number of PDO packets received.
SdoTxCount	40	2	DINT	The number of SDO packets transmitted.
SdoRxCount	42	2	DINT	The number of SDO packets received.
TimePcktCount	44	2	DINT	The number of TIME packets received or sent.
SyncPcktCount	46	2	DINT	The number of SYNC packets received or sent.
EmergencyPcktCount	48	2	DINT	The number of EMCY packets received or sent.
HeartbeatPcktCount	50	2	DINT	The number of Heartbeat packets received.
Reserved	52	2	DINT	This value is reserved.
Reserved	54	2	DINT	This value is reserved.
Reserved	56	2	DINT	This value is reserved.
Reserved	58	2	DINT	This value is reserved.
SlaveMapTransCount	60	128	INT[128]	The transaction count for all mapped items in the Virtual Slave Map (when the module is operating as a CANopen Slave). This will increase each time the specific mapped item either transmits or receives the correct PDO.
Reserved	188	62	INT[62]	These values are reserved.

Table 6.15 – Modbus Master Status Regiser Format

6.3.2.2. SLAVE DEVICE STATUS REGISTER

The Slave Device Status register will start at the configured offset (in the *Modbus Addressing* configuration) and each parameter will then have a HR/IR Offset relative to the starting address.

NOTE: The status information for CANopen Slave Node 1 to 124 will be provided in the Slave Device Mapping. Node 125 – 127 will **not** be shown in the Device Status Registers.

Field	HR/IR Offset	HR/IR Size (16bit words)	Data Type	Description
Slave Node 1 Node Address and Status	0	2	DINT	Node and Status for Slave Node Address 1

Node Address	*0.0 – 0.7*	-	*SINT*	*The node address of the mapped slave on the CANopen network.*
Reserved	*0.8 – 0.15*	-	*SINT*	*Reserved*
***Status**.Online*	*0.16*	-	*BOOL*	*When the last response received from the slave is less than the Slave Inactive Timeout parameter in the CAN Bus configuration, the slave is considered online, and this bit is set.*
***Status**.ErrRecv*	*0.17*	-	*BOOL*	*Set when the last EMCY message received from the slave has an error.*
***Status**.PDOErr*	*0.18*	-	*BOOL*	*Set if one of the PDOs are not operating correctly.*
***Status**.Initializing*	*0.19*	-	*BOOL*	*Set when the slave is in the initialize state.*
***Status**.Stopped*	*0.20*	-	*BOOL*	*Set when the slave is in the stopped state.*
***Status**.Operational*	*0.21*	-	*BOOL*	*Set when the slave is in the operational state.*
***Status**.PreOperational*	*0.22*	-	*BOOL*	*Set when the slave is in the pre- operational state.*
***Status**.Reserved*	*0.23*	-	*BOOL*	*N/A*
***Status**.PDOMapOk*	*0.24*	-	*BOOL*	*Indication that all the PDOs exchanges are ok.*
***Status.**Reserved*	*0.25 – 0.31*	-	*BOOL*	*N/A*
Slave Node 2 Node Address and Status	2	2	DINT	Node and Status for Slave Node Address 2
Slave Node 3 Node Address and Status	4	2	DINT	Node and Status for Slave Node Address 3
…	…	…	…	…
Slave Node 124 Node Address and Status	246	2	DINT	Node and Status for Slave Node Address 124
Reserved	248	2	DINT	This value is reserved.

Table 6.16 – Modbus Slave Device Status Regiser Format

6.3.2.3. CONTROL REGISTER

The Master Control register will start at the configured offset (in the *Modbus Addressing* configuration) and each parameter will then have an HR Offset relative to the starting address.

NOTE: When the CANopen Router/B is a CANopen Master then the CANopen network state and operation can be controlled in the Master Control Registers.

NOTE: When the CANopen Router/B is a CANopen Slave and a TPDO transmission type in the Virtual Slave map is set to *Evt-Interface* then the emulated TDPOs in the Virtual Slave map can be triggered for transmission by toggling the bit values in the *SlaveModeTPDOOutputTrig* field.

Field	HR Offset	HR Size (16bit words)	Data Type	Description
Master Control	0	2	DINT	Network Control Bits
NetworkPreOperational	*0.0*	-	*SINT*	When the CANopen Router is the CANopen Master, this bit will force the CANopen network to be PreOperational. **NOTE:** When other NetworkPreOperational and NetworkStop bits are not set then the CANopen Router will set the network state to Operational.
NetworkStop	*0.1*	-	*SINT*	When the CANopen Router is the CANopen Master, this bit will force the CANopen network to be Stopped. **NOTE:** When other NetworkPreOperational and NetworkStop bits are not set then the CANopen Router will set the network state to Operational.
Inhibit	*0.2*	-	*BOOL*	Inhibit the CANopen communication.
Reserved	*0.3 – 0.31*	-	*BOOL*	*N/A*
UTC	2	2	DINT[2]	When the CANopen Router is a CANopen Master, the user can write the Logix WallClock time to the UTC tag which will be converted into the CANopen time format for when sending TIME messages.
SlaveModeTPDOOutputTrig	6	8	BOOL[128]	When the CANopen Router is operating as a CANopen Slave, these bits are used to trigger sending of TPDOs when the Transmission Type is Evt – Interface. Each time the bit is toggle (either from 1 to 0 or from 0 to 1) the respective PDO will send the data to the CANopen Master.

Table 6.17 – Modbus Master Control Format

6.3.2.4. TPDO TRIGGER REGISTER

The TPDO Trigger Control register will start at the configured offset (in the *Modbus Addressing* configuration) and each parameter will then have a HR Offset relative to the starting address.

NOTE: The TPDO information for CANopen Slave Node 1 to 124 will be provided in the Device TPDO Trigger Registers. Node 125 – 127 will **not** be available in the Device TPDO Trigger Registers.

NOTE: When the CANopen Router/B is a CANopen Master and the CANopen Slave device RPDO (Tx) transmission type has been set to *Evt-Interface*, then the respective PDO can be triggered for transmission by changing the bit values in the TPDO trigger Registers.

Field	HR Offset	HR Size (16bit words)	Data Type	Description
Slave Node 1 TPDO Trigger Ctrl	0	2	DINT	Slave Node Address 1 TPDO Trigger Control
TPDO 0 Transmit Trigger	*0.0*	-	*BOOL*	*TPDO Index 0 Trigger*
TPDO 1 Transmit Trigger	*0.1*	-	*BOOL*	*TPDO Index 1 Trigger*
…	…	…	…	…
TPDO 31 Transmit Trigger	*0.31*	-	*BOOL*	*TPDO Index 31 Trigger*
Slave Node 2 TPDO Trigger Ctrl	2	2	DINT	Slave Node Address 2 TPDO Trigger Control
Slave Node 3 TPDO Trigger Ctrl	4	2	DINT	Slave Node Address 3 TPDO Trigger Control
…	…	…	…	…
Slave Node 124 TPDO Trigger Ctrl	246	2	DINT	Slave Node Address 124 TPDO Trigger Control

Table 6.18 – Modbus Device TPDO Trigger Control Format

6.4. MODBUS SLAVE

6.4.1. OPERATION

When the CANopen Router/B module's primary interface is set to Modbus Slave, then the CANopen data (PDO or SDO) can be mapped to and from configurable internal Modbus Registers and offsets.

The internal Modbus Registers can then be asynchronously exchanged with a remote Modbus Master on any of the physical ports (Ethernet TCP, RS232, or RS485). The remote Modbus Master can read or write to the configured Modbus addresses to access the CANopen data.

In the example below the CANopen Router (as a CANopen Master) with the primary interface set to Modbus Slave will have multiple Modbus Holding Registers written from a Modbus Master and then transmit that data to a CANopen Slave device.

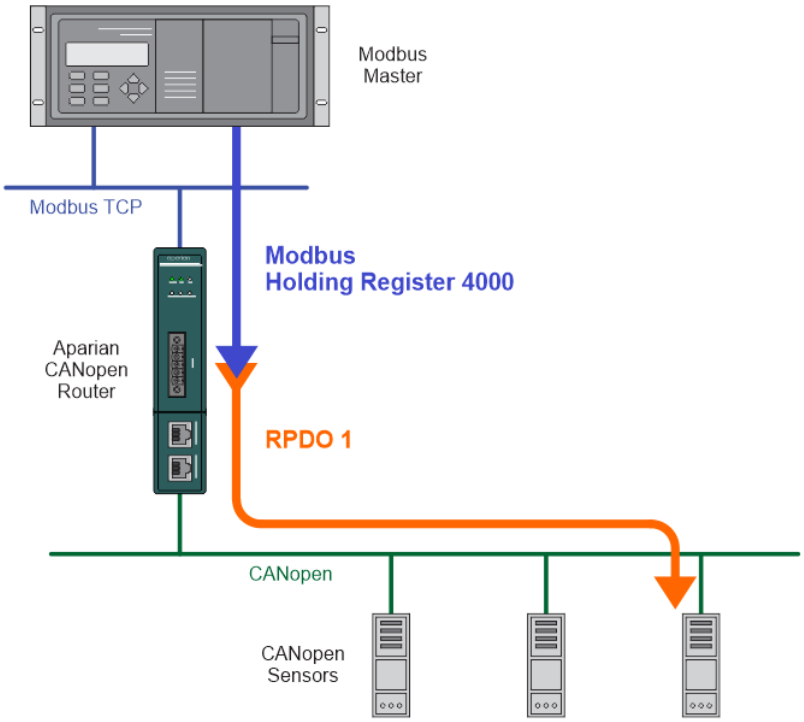


Figure 6.17 – Modbus Slave to CANopen Slave operation

For this example the remote Modbus Master will write data to Modbus Holding Register 4000 in the CANopen Router/B. Next the RPDO (Tx) for the slave device needs to be mapped to the internal Modbus Holding Register (offset 4000) which is where the data received from the Modbus Slave was stored.

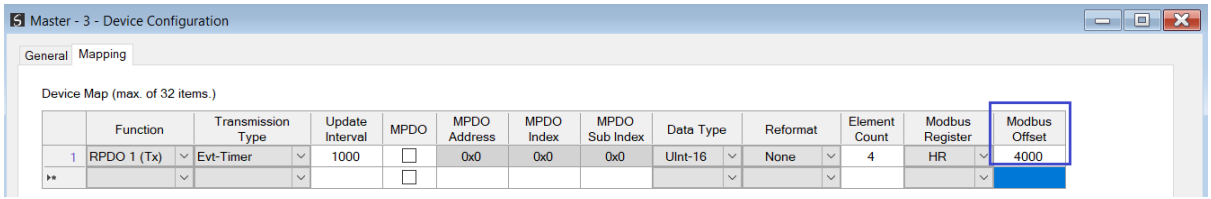


Figure 6.18 – CANopen Slave Modbus Mapping

NOTE: Every PDO can consume four Modbus Holding Registers, because the max PDO size is 8 bytes which equates to 4 Modbus words.

NOTE: The user will need to ensure that when writing to the CANopen Router Modbus Holding Registers that the registers holding data from the device are not inadvertently overwritten.

6.4.2. FIXED MODBUS MAPPING

When the CANopen Router/B is operating as a Modbus Slave it will provide status and control via configurable Modbus Registers.

The Master Status and Slave Device can be written to either Modbus Holding Register or Modbus Input Register at a configured offset. The Status data will be populated by the CANopen Router/B module.

The Control and TPDO Trigger data will be read from a the configured Modbus Holding Registers. The Control and Trigger data will need to be populated by the remote Modbus device.

NOTE: When the CANopen Router/B is a CANopen Master then the CANopen network state and operation can be controlled in the Master Control Registers.

NOTE: When the CANopen Router/B is a CANopen Slave and a TPDO transmission type in the Virtual Slave map is set to *Evt-Interface* then the emulated TDPOs in the Virtual Slave map can be triggered for transmission by toggling the bit values in the *SlaveModeTPDOOutputTrig* field.

NOTE: When the CANopen Router/B is a CANopen Master and the CANopen Slave device RPDO (Tx) transmission type has been set to *Evt-Interface*, then the respective PDO can be triggered for transmission by toggling the bit values in the TPDO trigger Registers.

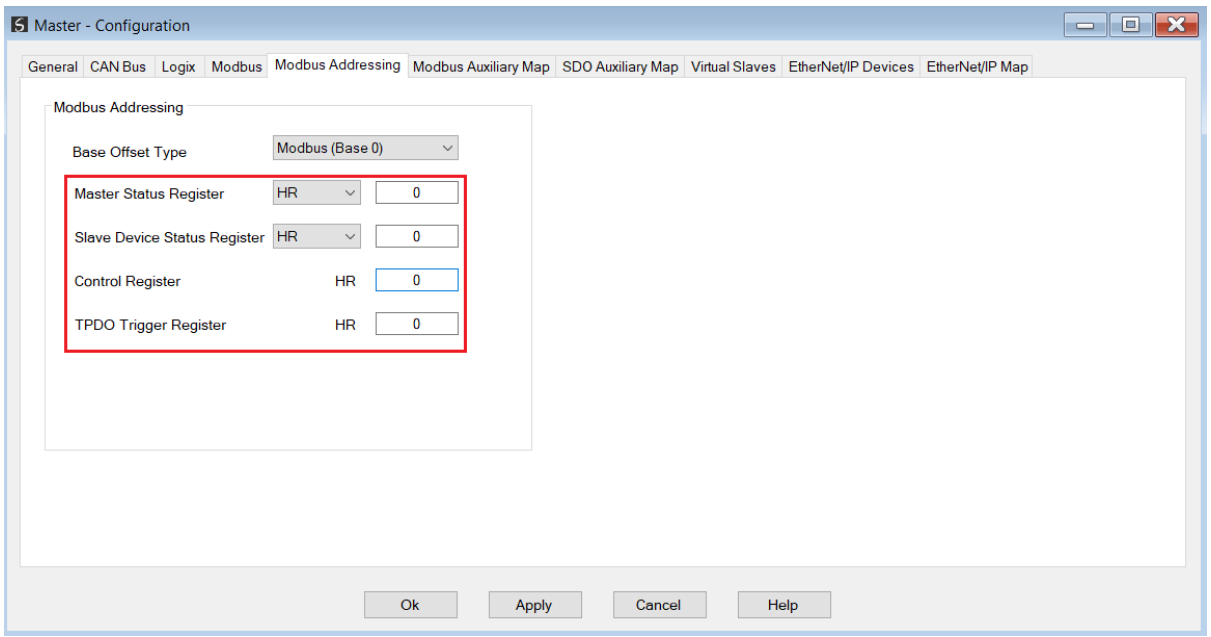


Figure 6.19 – Modbus Status and Control Addressing

A summary of the utilized Modbus Registers can be viewed by right-clicking on the module and selecting the ***Export Modbus Report*** option. This produces a CSV (comma-separated-variable) file showing the Modbus registers used.

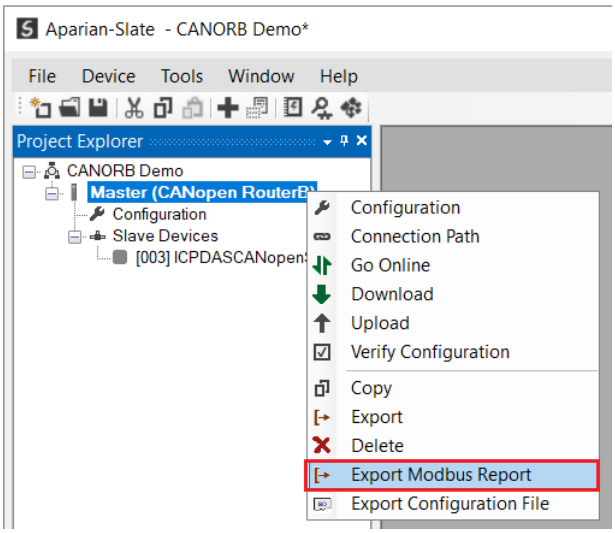


Figure 6.20 – Export Modbus Report

Below is the format of each of the fixed Modbus Registers.

6.4.2.1. MASTER STATUS REGISTER

The Master Status register will start at the configured offset (in the *Modbus Addressing* configuration) and each parameter will then have a HR/IR Offset relative to the starting address.

Field	HR/IR Offset	HR/IR Size (16bit words)	Data Type	Description
InstanceNameLen	0	2	DINT	The length of the instance name of the module that was configured under the general CANopen Router/B configuration in Slate.
InstanceName	2	8	SINT[16]	This instance name of the module that was configured under the general CANopen Router/B configuration in Slate.
Status	**10**	**2**	**DINT**	**Status Flags in Boolean Format**
***Status**.ConfigValid*	*10.0*	-	*BOOL*	*Set if a valid configuration is executing in the module.*

***Status**.DuplicateNode*	*10.1*	-	*BOOL*	*Set if a duplicate node is detected on the network.*
***Status**.NetworkOperational*	*10.2*	-	*BOOL*	*The current state of the CANopen network is operational.*
***Status**.NetworkPreOperational*	*10.3*	-	*BOOL*	*The current state of the CANopen network is pre-operational.*
***Status**.NetworkStopped*	*10.4*	-	*BOOL*	*The current state of the CANopen network is stopped.*
***Status**.MasterMode*	*10.5*	-	*BOOL*	*The CANopen Router is operating as a CANopen Master.*
***Status**.SlaveMode*	*10.6*	-	*BOOL*	*The CANopen Router is operating as a CANopen Slave.*
***Status**.MBCommsOnline*	*10.7*	-	*BOOL*	*The Modbus communication (in Modbus Master or Modbus Slave mode) is ok.* *In Master Master mode, if any of the mapped items in the Modbus Auxiliary Map has failed then this bit will be cleared.* *In Modbus Slave mode, if Modbus communication has not been received within the Modbus Inactivity Timeout then this bit will be cleared.*
***Status**.CANCommsInhibited*	*10.8*	-	*BOOL*	*Module CANopen communication has been inhibited.*
***Status**.BottomPwr*	*10.9*	-	*BOOL*	*Module is receiving power from the bottom power connector.*
***Status**.FrontPwr*	*10.10*	-	*BOOL*	*Module is receiving power from the front CAN connector.*
***Status**.EIPCommsOk*	*10.11*	-	*BOOL*	*The EtherNet/IP communication (in EtherNet/IP Target or EtherNet/IP Originator mode) is ok.* *In EtherNet/IP Target mode, if any of the Logix tag exchanges has failed or if the Class 1 connection has been lost, then this bit will be cleared.* *In EtherNet/IP Originator mode, if any of the mapped items in the EtherNet/IP Explicit Message Map has failed or if any of the Class 1 connections are not active, then this bit will be cleared.*
***Status**.Reserved*	*10.12-10.31*	-	*BOOL*	*N/A*
TransactionRate	12	2	DINT	The transaction rate is the number of CANopen messages per second that the module is currently routing.

DeviceTemperature	14	2	REAL	The internal temperature of the CANopen Router module.
UTCTime	16	4	DINT[2]	The UTC time on the CANopen network. This has already been formatted for Logix and can be viewed in LINT – Date/Time format.
RxCANCount	20	2	DINT	Received CAN message count.
TxCANCount	22	2	DINT	Transmitted CAN message count.
CanCrcErrCount	24	2	DINT	CAN CRC failed message count.
CanBitErrCount	26	2	DINT	CAN Bit error count.
CanStuffErrCount	28	2	DINT	CAN Stuff error count.
CanBusOffCount	30	2	DINT	The number of times the CAN receiver has detected the Bus Off state.
CanAckErrCount	32	2	DINT	The number of times the CAN message was no acknowledged.
CanFormatErrCount	34	2	DINT	The number of time a fixed format part of the received frame has the wrong format.
PdoTxCount	36	2	DINT	The number of PDO packets transmitted.
PdoRxCount	38	2	DINT	The number of PDO packets received.
SdoTxCount	40	2	DINT	The number of SDO packets transmitted.
SdoRxCount	42	2	DINT	The number of SDO packets received.
TimePcktCount	44	2	DINT	The number of TIME packets received or sent.
SyncPcktCount	46	2	DINT	The number of SYNC packets received or sent.
EmergencyPcktCount	48	2	DINT	The number of EMCY packets received or sent.
HeartbeatPcktCount	50	2	DINT	The number of Heartbeat packets received.
Reserved	52	2	DINT	This value is reserved.
Reserved	54	2	DINT	This value is reserved.
Reserved	56	2	DINT	This value is reserved.
Reserved	58	2	DINT	This value is reserved.
SlaveMapTransCount	60	128	INT[128]	The transaction count for all mapped items in the Virtual Slave Map (when the module is operating as a CANopen Slave). This will increase each time the specific mapped item either transmits or receives the correct PDO.
Reserved	188	62	INT[62]	These values are reserved.

Table 6.19 – Modbus Master Status Regiser Format

6.4.2.2. SLAVE DEVICE STATUS REGISTER

The Slave Device Status register will start at the configured offset (in the *Modbus Addressing* configuration) and each parameter will then have a HR/IR Offset relative to the starting address.

NOTE: The status information for CANopen Slave Node 1 to 124 will be provided in the Slave Device Mapping. Node 125 – 127 will **not** be shown in the Device Status Registers.

Field	HR/IR Offset	HR/IR Size (16bit words)	Data Type	Description
Slave Node 1 Node Address and Status	0	2	DINT	Node and Status for Slave Node Address 1
Node Address	*0.0 – 0.7*	-	*SINT*	*The node address of the mapped slave on the CANopen network.*
Reserved	*0.8 – 0.15*	-	*SINT*	*Reserved*
***Status**.Online*	*0.16*	-	*BOOL*	*When the last response received from the slave is less than the Slave Inactive Timeout parameter in the CAN Bus configuration, the slave is considered online, and this bit is set.*
***Status**.ErrRecv*	*0.17*	-	*BOOL*	*Set when the last EMCY message received from the slave has an error.*
***Status**.PDOErr*	*0.18*	-	*BOOL*	*Set if one of the PDOs are not operating correctly.*
***Status**.Initializing*	*0.19*	-	*BOOL*	*Set when the slave is in the initialize state.*
***Status**.Stopped*	*0.20*	-	*BOOL*	*Set when the slave is in the stopped state.*
***Status**.Operational*	*0.21*	-	*BOOL*	*Set when the slave is in the operational state.*
***Status**.PreOperational*	*0.22*	-	*BOOL*	*Set when the slave is in the pre- operational state.*
***Status**.Reserved*	*0.23*	-	*BOOL*	*N/A*
***Status**.PDOMapOk*	*0.24*	-	*BOOL*	*Indication that all the PDOs exchanges are ok.*
***Status.**Reserved*	*0.25 – 0.31*	-	*BOOL*	*N/A*
Slave Node 2 Node Address and Status	2	2	DINT	Node and Status for Slave Node Address 2
Slave Node 3 Node Address and Status	4	2	DINT	Node and Status for Slave Node Address 3

…	…	…	…	…
Slave Node 124 Node Address and Status	246	2	DINT	Node and Status for Slave Node Address 124
Reserved	248	2	DINT	This value is reserved.

Table 6.20 – Modbus Slave Device Status Regiser Format

6.4.2.3. CONTROL REGISTER

The Master Control register will start at the configured offset (in the *Modbus Addressing* configuration) and each parameter will then have a HR Offset relative to the starting address.

NOTE: When the CANopen Router/B is a CANopen Master then the CANopen network state and operation can be controlled in the Master Control Registers.

NOTE: When the CANopen Router/B is a CANopen Slave and a TPDO transmission type in the Virtual Slave map is set to *Evt-Interface* then the emulated TDPOs in the Virtual Slave map can be triggered for transmission by toggling the bit values in the *SlaveModeTPDOOutputTrig* field.

Field	HR Offset	HR Size (16bit words)	Data Type	Description
Master Control	0	2	DINT	Network Control Bits
NetworkPreOperational	*0.0*	-	*SINT*	When the CANopen Router is the CANopen Master, this bit will force the CANopen network to be PreOperational. **NOTE:** When other NetworkPreOperational and NetworkStop bits are not set then the CANopen Router will set the network state to Operational.
NetworkStop	*0.1*	-	*SINT*	When the CANopen Router is the CANopen Master, this bit will force the CANopen network to be Stopped. **NOTE:** When other NetworkPreOperational and NetworkStop bits are not set then the CANopen Router will set the network state to Operational.
Inhibit	*0.2*	-	*BOOL*	Inhibit the CANopen communication.
Reserved	*0.3 – 0.31*	-	*BOOL*	*N/A*

UTC	2	2	DINT[2]	When the CANopen Router is a CANopen Master, the user can write the Logix WallClock time to the UTC tag which will be converted into the CANopen time format for when sending TIME messages.
SlaveModeTPDOOutputTrig	6	8	BOOL[128]	When the CANopen Router is operating as a CANopen Slave, these bits are used to trigger sending of TPDOs when the Transmission Type is Evt – Interface. Each time the bit is toggle (either from 1 to 0 or from 0 to 1) the respective PDO will send the data to the CANopen Master.

Table 6.21 – Modbus Master Control Format

6.4.2.4. TPDO TRIGGER REGISTER

The TPDO Trigger Control register will start at the configured offset (in the *Modbus Addressing* configuration) and each parameter will then have a HR Offset relative to the starting address.

NOTE: The TPDO information for CANopen Slave Node 1 to 124 will be provided in the Device TPDO Trigger Registers. Node 125 – 127 will **not** be available in the Device TPDO Trigger Registers.

NOTE: When the CANopen Router/B is a CANopen Master and the CANopen Slave device RPDO (Tx) transmission type has been set to *Evt-Interface*, then the respective PDO can be triggered for transmission by changing the bit values in the TPDO trigger Registers.

Field	HR Offset	HR Size (16bit words)	Data Type	Description
Slave Node 1 TPDO Trigger Ctrl	0	2	DINT	Slave Node Address 1 TPDO Trigger Control
TPDO 0 Transmit Trigger	*0.0*	-	*BOOL*	*TPDO Index 0 Trigger*
TPDO 1 Transmit Trigger	*0.1*	-	*BOOL*	*TPDO Index 1 Trigger*
…	…	…	…	…
TPDO 31 Transmit Trigger	*0.31*	-	*BOOL*	*TPDO Index 31 Trigger*
Slave Node 2 TPDO Trigger Ctrl	2	2	DINT	Slave Node Address 2 TPDO Trigger Control
Slave Node 3 TPDO Trigger Ctrl	4	2	DINT	Slave Node Address 3 TPDO Trigger Control
…	…	…	…	…

Slave Node 124 TPDO Trigger Ctrl	246	2	DINT	Slave Node Address 124 TPDO Trigger Control

Table 6.22 – Modbus Device TPDO Trigger Control Format

6.5. CANOPEN SLAVE DEVICE

There are various controls for each CANopen Slave when the CANopen Router/B is operating as a CANopen Master. These controls are described below.

6.5.1. FORCE STATE

The operating state of a CANopen Slave can be forced to a specific state such that it does not follow the network operating state.

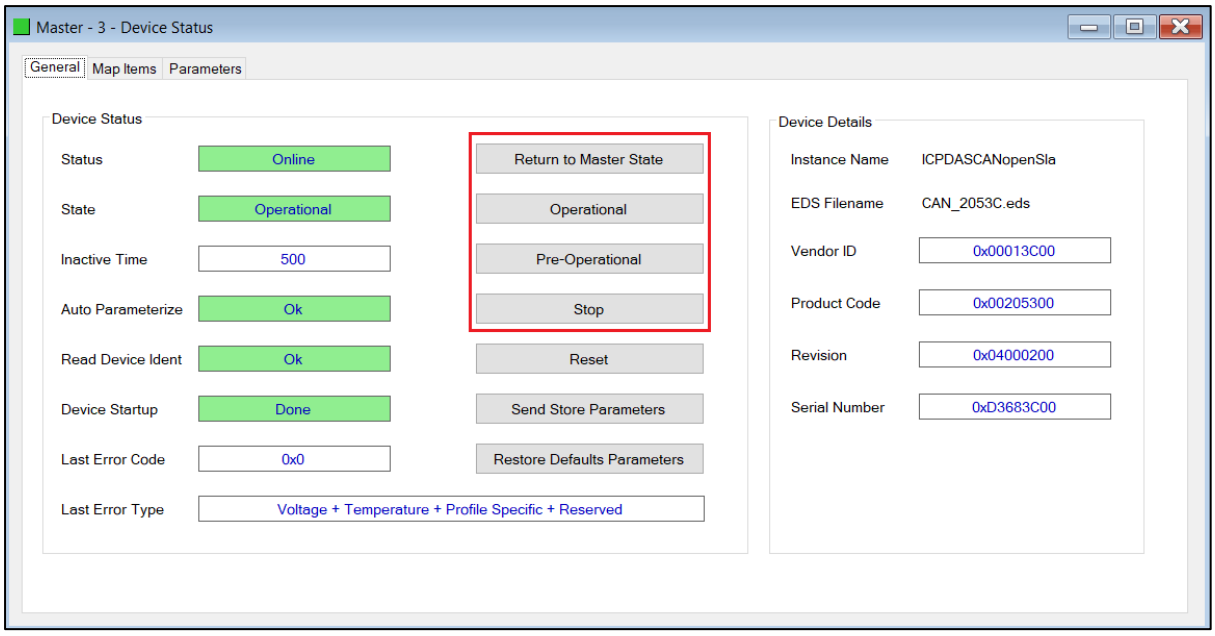


Figure 6.21 – CANopen Slave operating state command

By pressing *Operational, Pre-Operational,* or *Stop* will force the CANopen Slave device to go into that state.

- In Operational mode there will be Process and Service Data exchange (PDO and SDO) as well as Network Management messages accepted (NMT).
- In Pre-Operational mode there will only be Service Data exchange (SDO) as well as Network Management messages accepted (NMT).
- In Stopped mode there will be only be Network Management messages accepted (NMT).

Once the user presses the *Return to Master State* then the CANopen Slave device will return to the state of the CANopen network.

6.5.2. RESET

When pressing this button it will send a reset command to the selected CANopen device to perform a reset.

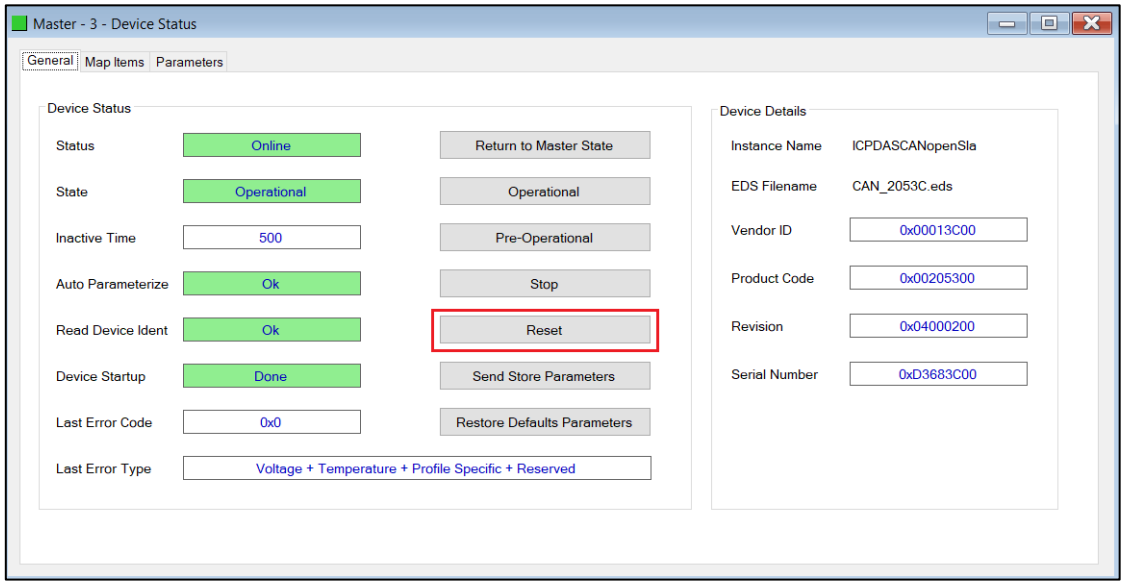


Figure 6.22 – CANopen Slave Reset command

6.5.3. STORE DEVICE PARAMETERS

The *Send Store Parameters* option is used to request the CANopen Slave device to save all the internal parameters (which is accessed through the *Parameters* tab in the Device Status window in Slate) to the CANopen Slave's internal Non-Volatile (NV) memory. This will allow the CANopen Slave to start-up with the last parameter changes made.

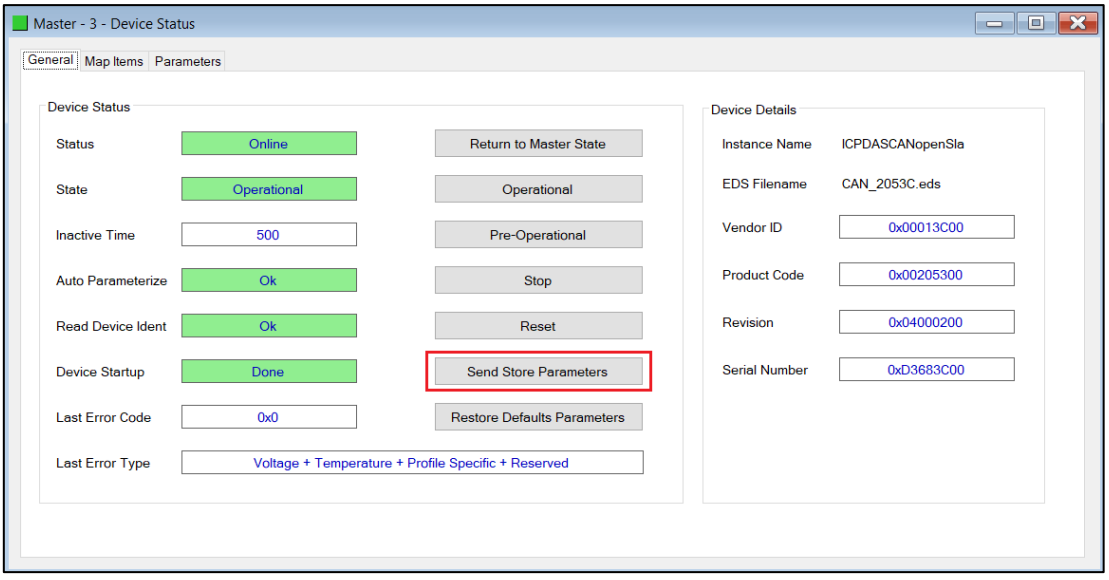


Figure 6.23 – CANopen Slave Store command

6.5.4. Restore Default Device Parameters

The *Restore Default Parameters* option is used to request the CANopen Slave device to revert all the internal parameters (which is accessed through the *Parameters* tab in the Device Status window in Slate) back to factory defaults.

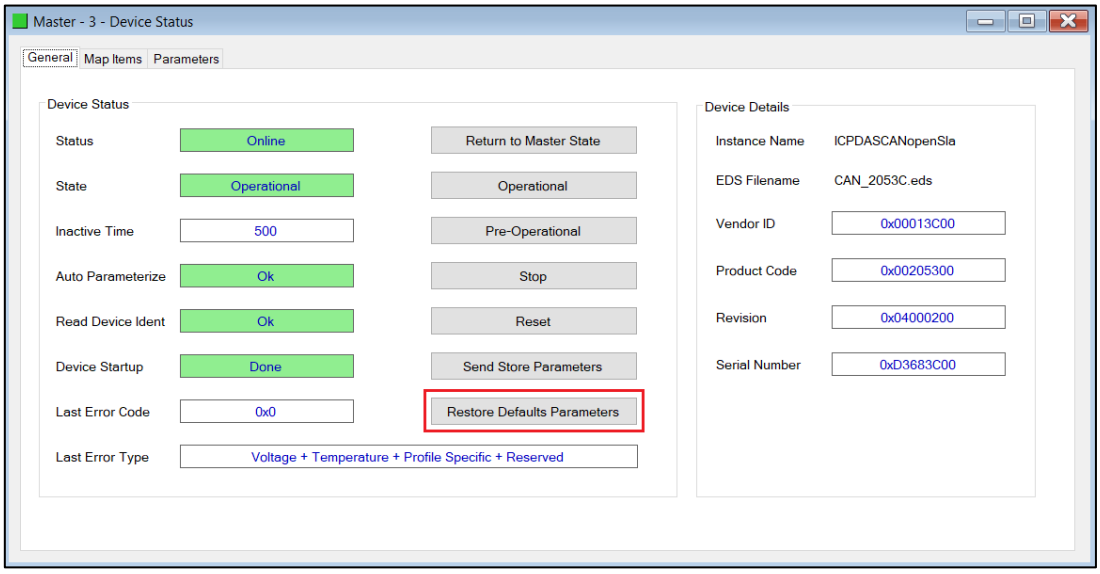


Figure 6.24 – CANopen Slave Restore Defaults command

6.5.5. Parameterization

The CANopen Slave device internal parameters can be accessed using the *Parameters* tab in the Device Status window in Slate. The parameter list is provided by EDS file selected when when adding the CANopen Slave device to the module Device tree.

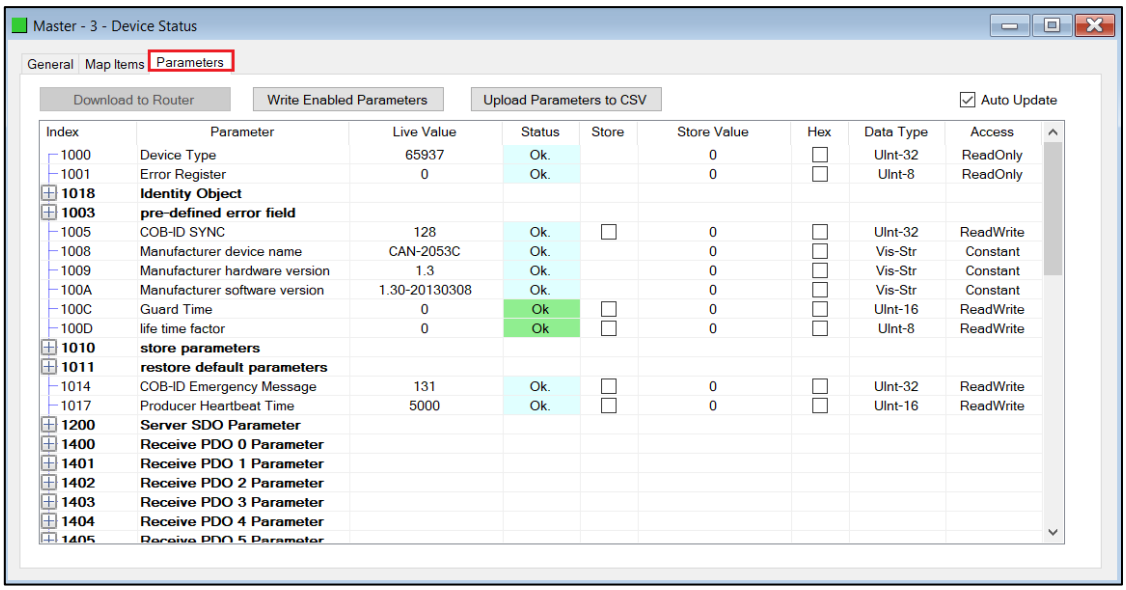


Figure 6.25 – CANopen Slave parameters

6.5.5.1. READ VALUES

If the user wants to read a value from a specific parameter, right-click on the parameter and select *Refresh* (as shown below). This will read the parameter from the CANopen Slave and update it in the *Live Value* column.

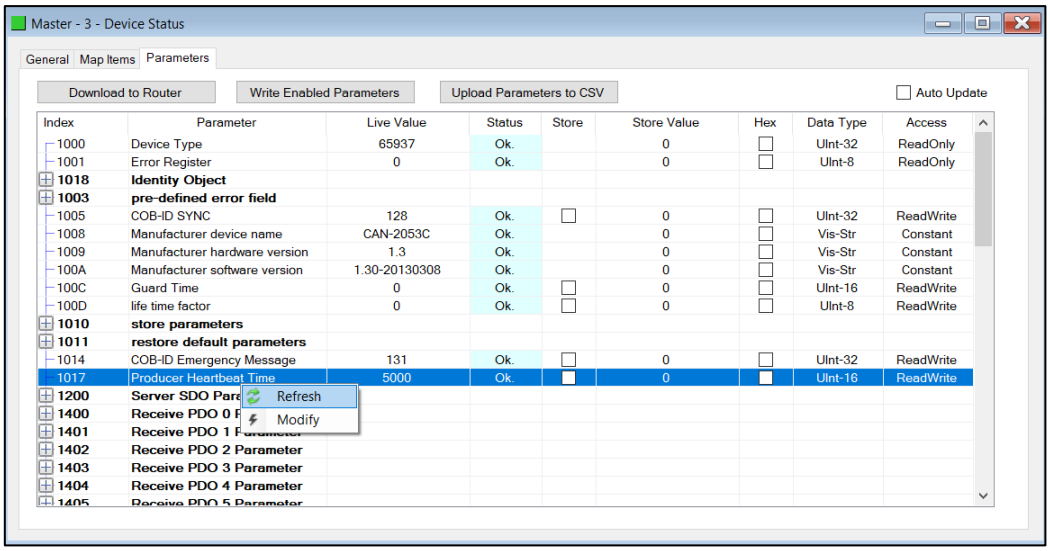


Figure 6.26 – Slave device parameters – Read

6.5.5.2. WRITE VALUES

When the Access to a device is *ReadWrite*, then the value can be modified by the user. This is done by either right-clicking on the value and selecting *Modify*, or double-clicking on the item. The user will be able to enter the new value into the textbox which will then be downloaded into the slave device.

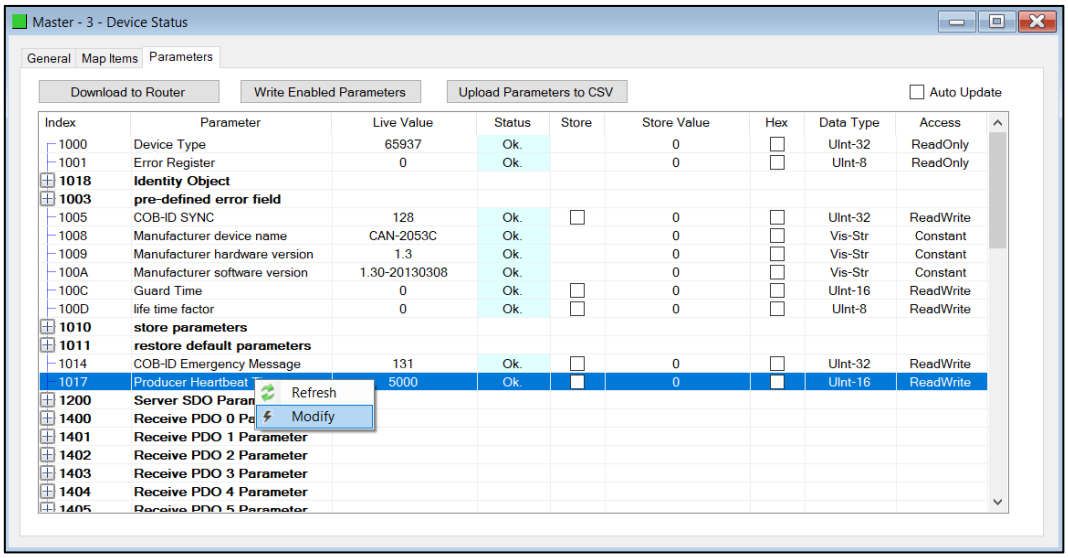


Figure 6.27 – Slave device parameters – Modify

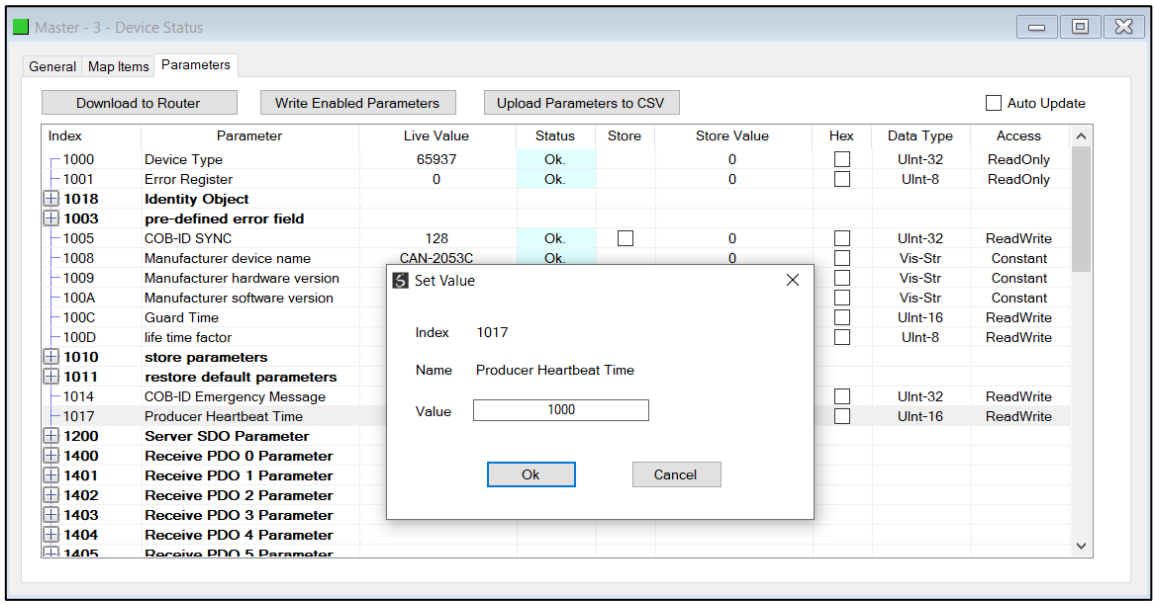


Figure 6.28 – Slave device parameters – Set Value

6.5.5.3. STORE PARAMETERS

NOTE: The parameters in the slave device will not automatically be saved to non-volatile memory. The user will need to write the required codes to the store parameters (parameter 1010) index to force the CANopen Slave device to store the updated parameters to NV memory. The user will require to write 0x65766173 to either 1010.1, 1010.2, or 1010.3 (as shown below). Alternatively, the user can use the *Send Store Parameters* in the Slave device status (see the *Diagnostics* or *Operations* section).

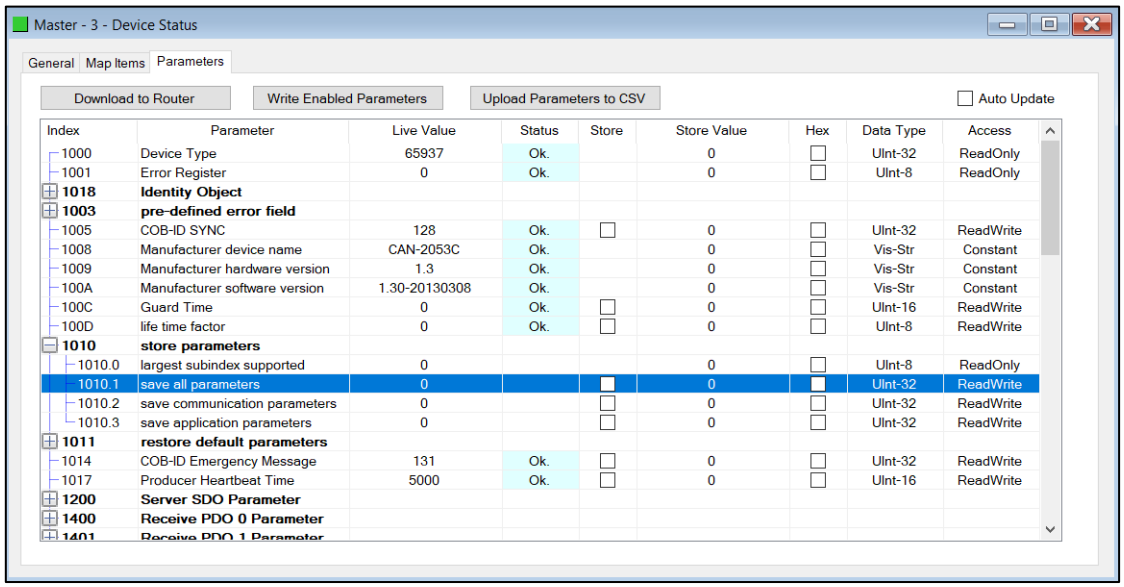


Figure 6.29 – Slave device parameters – Save Parameters

6.5.5.4. AUTO UPDATE

The user can also select to Auto-Update the values in the parameter list by selecting the auto-update checkbox (as shown below). Once this option is checked, all the visible parameter values will automatically start updating.

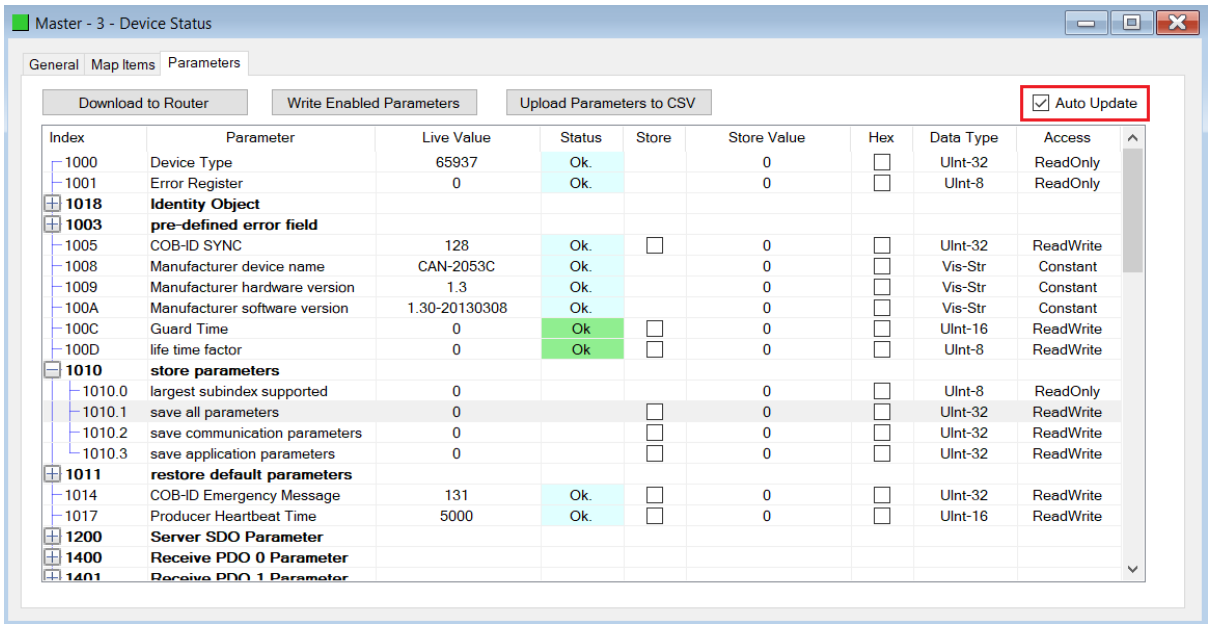


Figure 6.30 – Slave device parameters – Auto Update

6.5.5.5. DOWNLOAD TO ROUTER

The ***Download to Router*** function forces all changes made to the parameter ***Store*** and ***Store Value*** attributes to be written down to the CANopen Router/B.

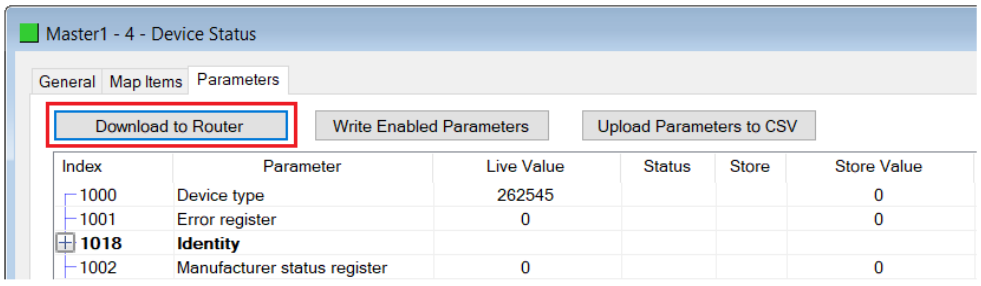


Figure 6.31 – Slave device parameters – Download to Router

NOTE: The ***Download to Router*** function forces a full configuration download to the CANopen Router/B module which will cause temporary disruption to the CANopen communication.

6.5.5.6. WRITE ENABLED PARAMETERS

The ***Write Enabled Parameters*** function will write the ***Store Value*** of each parameter with the ***Store*** attribute set, down to the slave device.

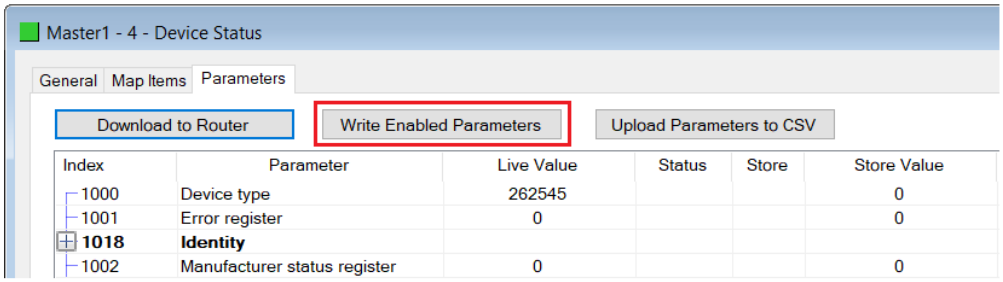


Figure 6.32 – Slave device parameters – Write Enabled Parameters

6.5.5.7. UPLOAD PARAMETERS TO CSV

The Upload Parameters to CSV function will first read all the parameter values from the slave device and then store them in a CSV (comma-separated-variable) file.

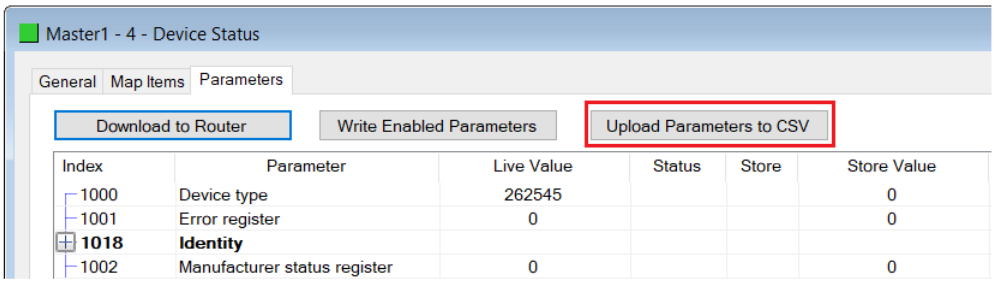


Figure 6.33 – Slave device parameters – Upload Parameters to CSV

7. DIAGNOSTICS

7.1. LEDS

The module provides six LEDs for diagnostics purposes as shown below. A description of each LED is given in the table below.

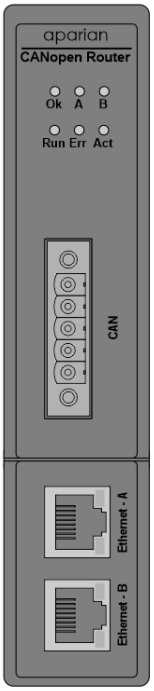


Figure 7.1 - CANopen Router/B front view

LED	Description
Ok	The module LED will provide information regarding the system-level operation of the module. If the LED is **red**, then the module is not operating correctly. For example, if the module application firmware has been corrupted or there is a hardware fault the module will have a red Module LED. If the LED is briefly **flashing red**, and then returning to either flashing green or solid green, then there is a duplicate IP address on the Ethernet network similar to that of the local module. If the LED is **green (flashing)**, then the module has booted and is running correctly **without** any application configuration loaded.

	If the LED is **green (solid)**, then the module has booted and is running correctly **with** application configuration loaded.
A / B	The Ethernet LED will light up when an Ethernet link has been detected (by plugging in a connected Ethernet cable). The LED will flash every time traffic is detected. This module has two Ethernet ports A and B. Each LEDs represents each specific port.
Run	The module Run LED will provide information regarding the operational state of the CANopen network. **Solid Green** – CANopen network is operational **Flashing Green** – CANopen network is pre-operational **Blink Green** – CANopen network is stopped
Err	The Err LED will provide information regarding the operational condition of the CANopen devices. **CANopen Master** **Solid Red** – No configuration has been loaded on the CANopen Router. **Flashing Red** – The primary interface (EtherNet/IP or Modbus) to the CANopen Router is not available. **Blink Red** – There is an issue with at least one CANopen Slave device. **Off** – There are no issues. **CANopen Slave** **Solid Red** – No configuration has been loaded on the CANopen Router. **Flashing Red** – The primary interface (EtherNet/IP or Modbus) to the CANopen Router is not available. **Blink Red** – There is an issue with at least one PDO in the CANopen Router when operating as a CANopen Slave device. **Off** – There are no issues.
Act	The activity LED is used for the activity on the Primary Interface (e.g. EtherNet/IP or Modbus). Every time a valid packet is received from the Primary Interface the LED will toggle green. The LED will toggle red if a corrupted packet was received (e.g. failed checksum when using RS232 or RS485).

Table 7.1 - Module LED operation

7.2. MODULE STATUS MONITORING IN SLATE

The CANopen Router/B can provide a range of statistics which can assist with module operation, maintenance, and fault finding. The statistics can be accessed in full by Slate or using the web server in the module.

To view the module's status in the Aparian-Slate environment, the module must be online. If the module is not already Online (following a recent configuration download), then right-click on the module and select the *Go Online* option.

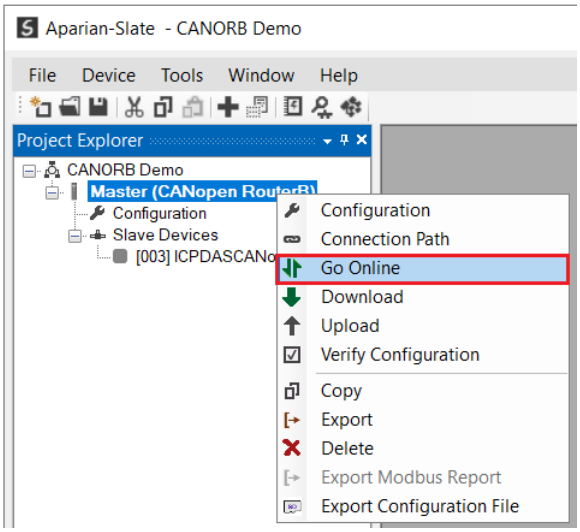


Figure 7.2. - Selecting to Go Online

The Online mode is indicated by the green circle behind the module in the Project Explorer tree.

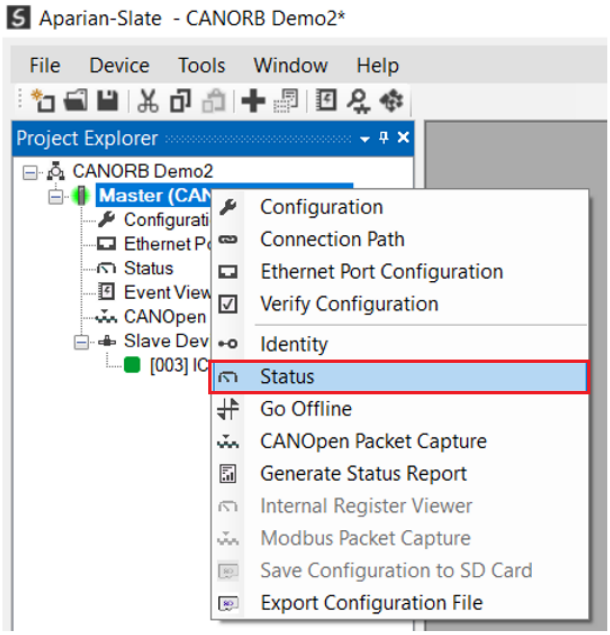


Figure 7.3. - Selecting online Status

The Status monitoring window can be opened by either double-clicking on the *Status* item in the Project Explorer tree, or by right-clicking on the module and selecting *Status*.

The status window contains multiple tabs to display the status of the module. Most of these parameters in the status windows are self-explanatory or have been discussed in previous sections.

7.2.1. GENERAL

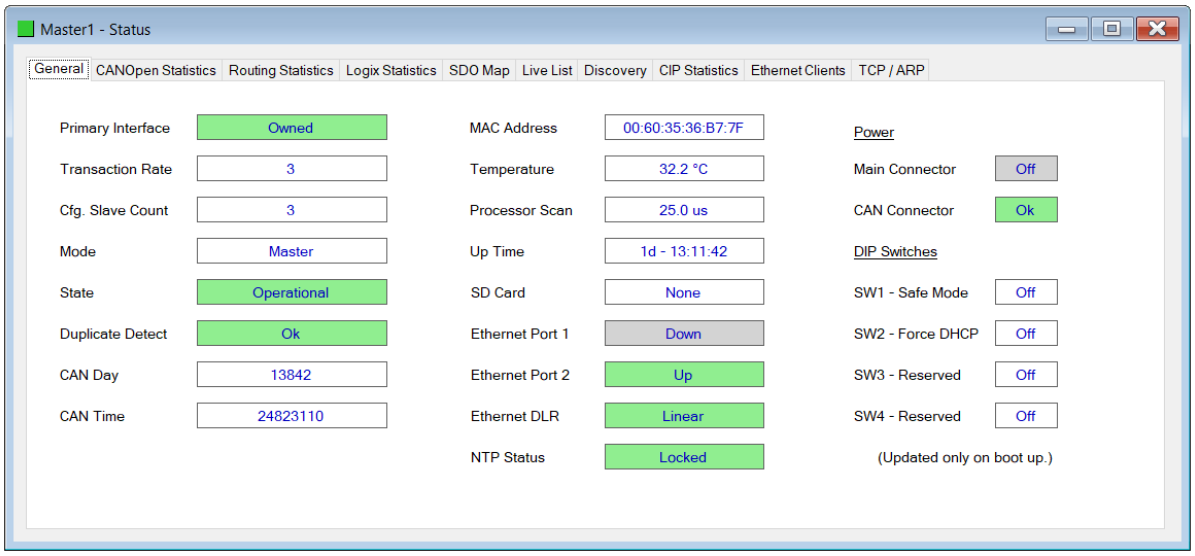


Figure 7.4. - Status monitoring - General

The General tab displays the following general parameters:

Parameter	Description
Primary Interface	The primary interface that was selected (EtherNet/IP or Modbus) and if the interface is online.
Transaction Rate	The transaction rate is the number of CANopen PDOs per second that the module is currently routing.
Cfg. Slave Count	The number of CANopen Slave devices that have been configured.
Mode	The CANopen Router can either be configured as a CANopen Master or CANopen Slave.
State	The operational state of the CANopen network. **Operational** In Operational mode there will be Process and Service Data exchange (PDO and SDO) as well as Network Management messages accepted (NMT). **Pre-operational** In Pre-Operational mode there will only be Service Data exchange (SDO) as well as Network Management messages accepted (NMT). **Stopped**

	In Stopped mode there will be only be Network Management messages accepted (NMT).
Duplicate Detect	Indicates if there is a duplicate node (same as the local node) on the CANopen network.
CAN Day	Current CAN Day (based on the Time on the CANopen network)
CAN Time	Current CAN Time (based on the Time on the CANopen network)
MAC Address	Displays the module's unique Ethernet MAC address.
Temperature	The internal temperature of the module.
Processor Scan	The amount of time (microseconds) taken by the module's processor in the last scan.
Up Time	Indicates the elapsed time since the module was powered-up.
SD Card	Indicates if a SD Card is present or not.
Ethernet Port 1 / 2	This is the status of each Ethernet port. **Down** The Ethernet connector has not been successfully connected to an Ethernet network. **Up** The Ethernet connector has successfully connected to an Ethernet network. **Mirror Enabled** The Ethernet port is mirroring the traffic on the other Ethernet port.
Ethernet DLR	The status of the Ethernet Device Level Ring (DLR). **Disabled** Device Level Ring functionality has been disabled. **Linear** The DLR functionality has been enabled and the Ethernet network architecture is linear. **Ring – Fault** The DLR functionality has been enabled and the Ethernet network architecture is ring, but there is a fault with the network. **Ring – Ok** The DLR functionality has been enabled and the Ethernet network architecture is ring and is operating as expected.
NTP Status	The status of the local NTP Client. **Disabled** The NTP time synchronization has been disabled. **Locked**

	NTP time synchronization has been enabled and the module has locked onto the target time server. **Not Locked** NTP time synchronization has been enabled and the module has not locked onto the target time server.
Power	Indication from which port the module is receiving power. **Main Connector** The power is plugged in at the bottom connector. **CAN Connector** The power is plugged in at the CAN connector.
DIP Switch Position	The status of the DIP switches when the module booted. **NOTE**: This status will not change if the DIP switches are altered when the module is running.

Table 7.2 - Parameters displayed in the Status Monitoring – General Tab

7.2.2. CANopen and CAN Statistics

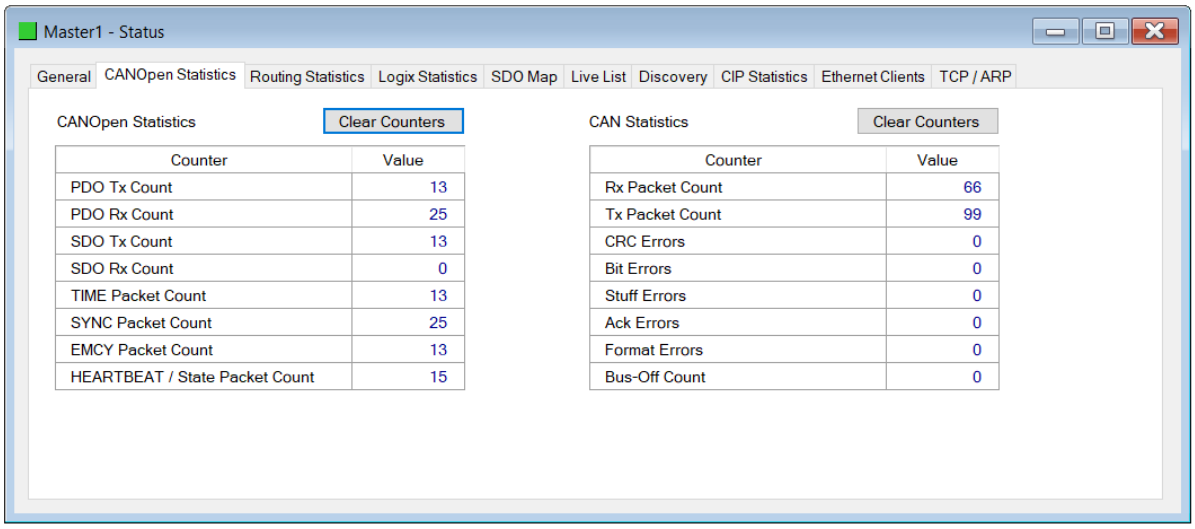


Figure 7.5. - Status monitoring – CANopen Statistics

The CANopen Statistics displays the following parameters:

Parameter	Description
PDO Tx Count	The number of PDO packets transmitted.
PDO Rx Count	The number of PDO packets received.
SDO Tx Count	The number of SDO packets transmitted.
SDO Rx Count	The number of SDO packets received.
TIME Packet Count	The number of TIME packets received or sent.

SYNC Packet Count	The number of SYNC packets received or sent.
EMCY Packet Count	The number of EMCY packets received or sent.
Heartbeat Packet Count	The number of Heartbeat packets received.

Table 7.3 - Parameters displayed in the Status Monitoring – CANopen Statistics Tab

The CAN Statistics displays the following parameters:

Parameter	Description
Rx CAN Packet Count	Received CAN message count.
Tx CAN Packet Count	Transmitted CAN message count.
CRC Errors	CAN CRC failed message count.
Bit Errors	CAN Bit error count.
Stuff Errors	CAN Stuff error count.
Ack Errors	The number of times the CAN message was no acknowledged.
Format Errors	The number of time a fixed format part of the received frame has the wrong format.
Bus-Off Count	The number of times the CAN receiver has detected the Bus Off state.

Table 7.4 - Parameters displayed in the Status Monitoring – CAN Statistics Tab

7.2.3. ROUTING STATISTICS

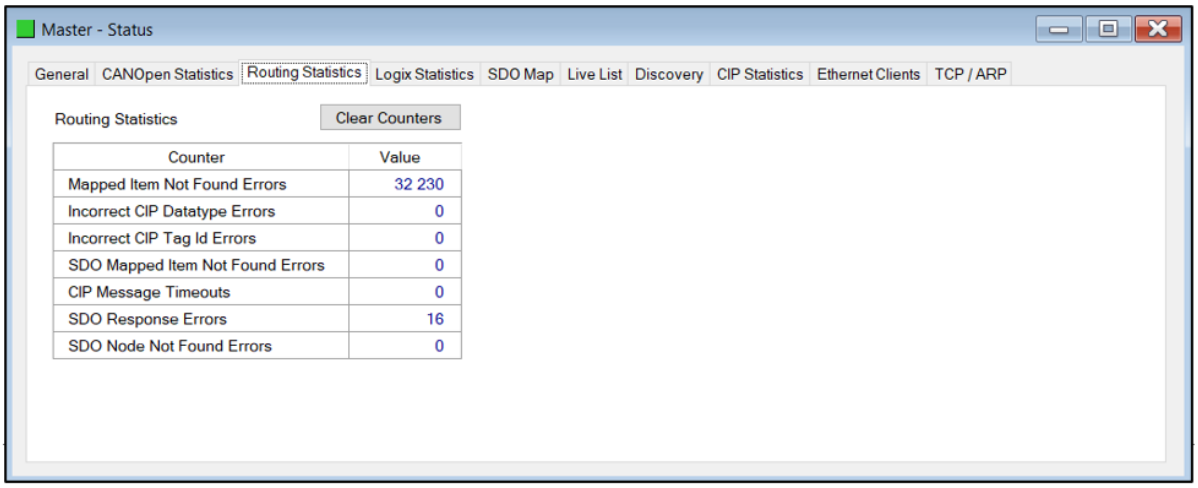


Figure 7.7. - Status monitoring – Logix Statistics

The Routing Statistics tab displays the following parameters:

Parameter	Description
Mapped Item Not Found Errors	A PDO was received from the CANopen network which has not been mapped in the CANopen Router/B.
Incorrect CIP Datatype Errors	The Logix tag data type has changed since the last update.
Incorrect CIP Tag Id Errors	The Logix UDT tag ID has changed since the last update.
SDO Mapped Item Not Found Errors	A SDO response was received from the CANopen network which does not have a matching request from the CANopen Router/B.
CIP Message Timeouts	The EtherNet/IP CIP message request has not received a response before the EIP timeout time.
SDO Response Errors	The SDO response received had an error code.
SDO Node Not Found Errors	The SDO response has a node that does not match the request.

Table 7.5 - Parameters displayed in the Status Monitoring – Routing Statistics Tab

7.2.4. VIRTUAL SLAVES

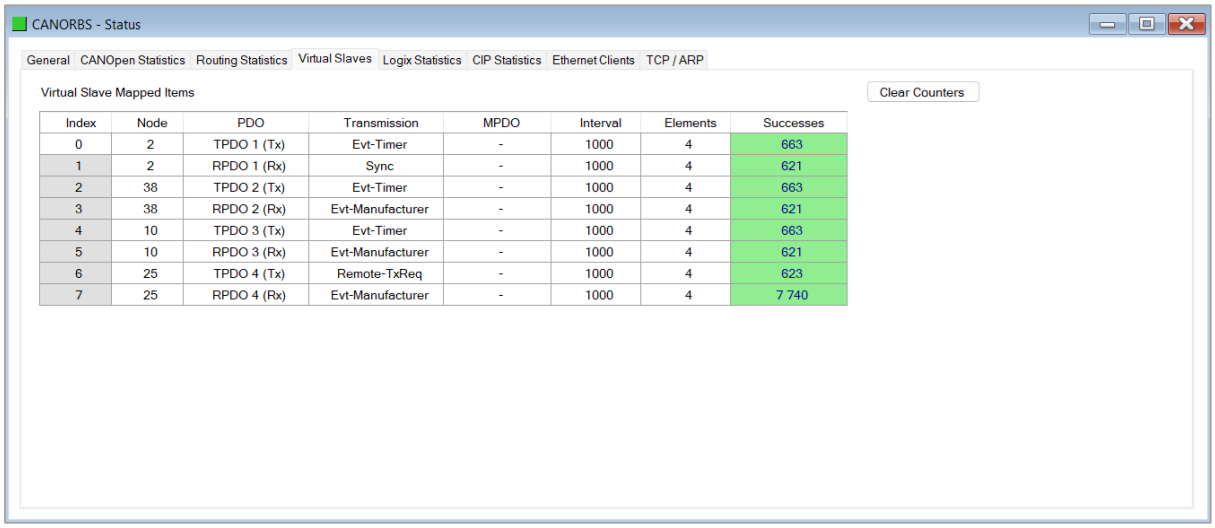


Figure 7.7. - Status monitoring – Virtual Slaves

The Virtual Slaves tab displays the following parameters:

Parameter	Description
Index	Index of the specific Virtual Slave mapped item.
Node	Node number of the specific Virtual Slave mapped item.
PDO	Process Data Object (PDO) of the specific Virtual Slave mapped item.

Transmission	The transmission type for the specific Virtual Slave mapped item.
MPDO	When used, the Multiplexed PDO (MPDO) for the specific Virtual Slave mapped item.
Interval	When relevant for the specific transmission type, the update interval (in milliseconds) for the specific Virtual Slave mapped item.
Elements	Number of elements for the specific Virtual Slave mapped item.
Successes	The successful transmission count for each specific Virtual Slave mapped item. The count will increase each time there is a successful transaction (receive or transmit) and will rollover at 65535.

Table 7.6 - Parameters displayed in the Status Monitoring – Virtual Slaves Tab

7.2.5. LOGIX STATISTICS

NOTE: The Logix statistics tab is only displayed if the module has the primary interface set to EtherNet/IP Target.

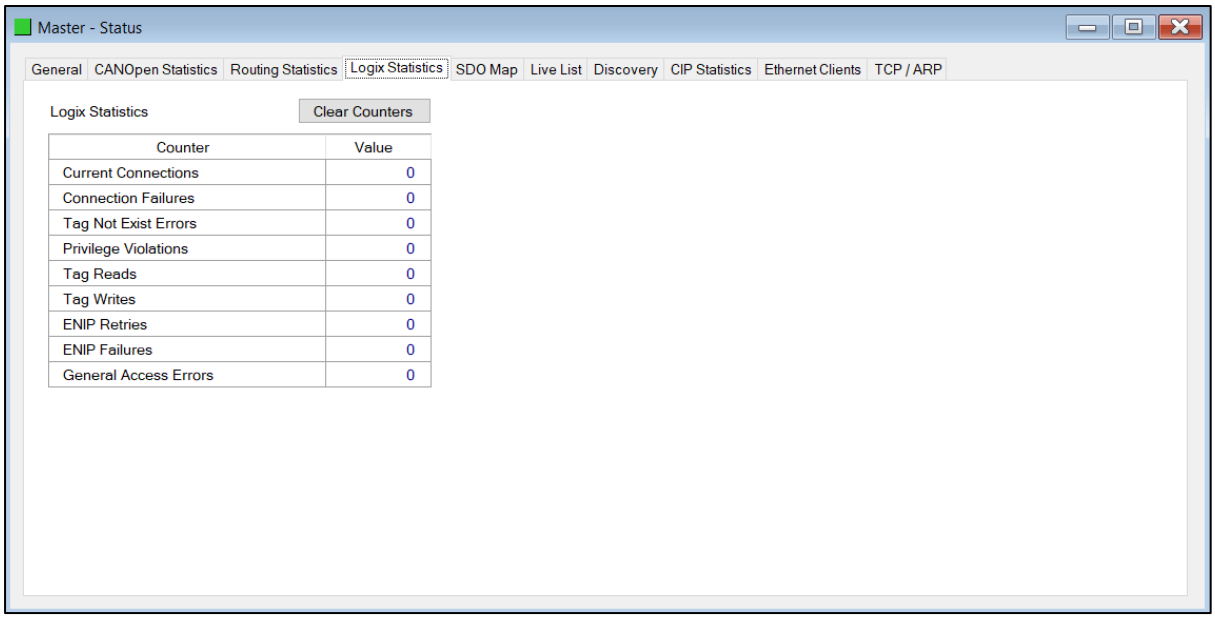


Figure 7.7. - Status monitoring – Logix Statistics

The Logix Statistics tab displays the following parameters:

Parameter	**Description**
Current Connections	The number of current open class 3 connections.
Connection Failures	The number of failed attempts at establishing a class 3 connection with a Logix controller.

Tag Not Exist Errors	The number of tag read and tag write transactions that failed due to the destination tag not existing.
Privilege Violations	The number of tag read and tag write transactions that failed due to a privilege violation error. This may be caused by the External Access property of the Logix tag being set to either None or Read Only.
Tag Reads	The number of tag read transactions executed by the CANopen Router/B module.
Tag Writes	The number of tag write transactions executed by the CANopen Router/B module.
ENIP Retries	This count increases when no response was received from the Logix Controller by the time the ENIP timeout is reached.
ENIP Failures	This count increases when the ENIP Retry Limit is reached and no response has been received from the Logix Controller.
Tag Access General Error	This count increases when a tag cannot be accessed for any other reason not reported above.

Table 7.7 - Parameters displayed in the Status Monitoring – Logix Statistics Tab

7.2.6. EtherNet/IP

NOTE: The EtherNet/IP tab is only displayed if the module has the primary interface set to EtherNet/IP Originator.

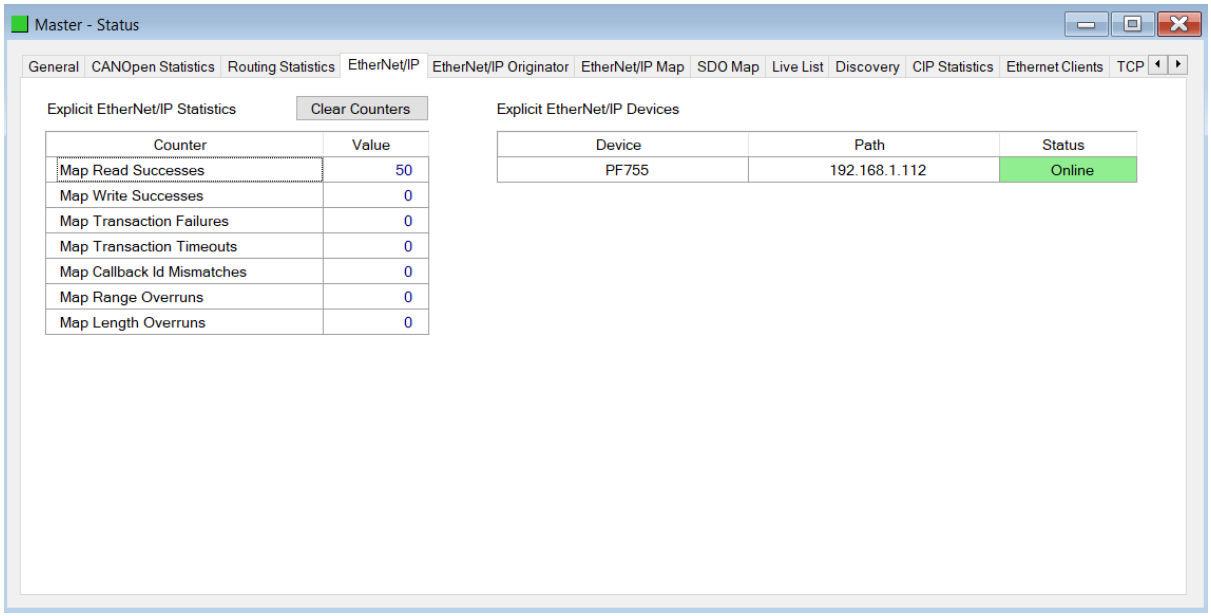


Figure 7.7. - Status monitoring – EtherNet/IP

The EtherNet/IP tab displays the following parameters:

Parameter	Description
Map Read Successes	A successful explicit read transaction has been completed in the EtherNet/IP Map.
Map Write Successes	A successful explicit write transaction has been completed in the EtherNet/IP Map.
Map Transaction Failures	A failed explicit transaction has occurred in the EtherNet/IP Map (e.g. error code was returned).
Map Transaction Timeouts	The explicit transaction in the EtherNet/IP Map has not received a response within the allowed timeout.
Map Callback Id Mismatches	The callback Id received for the EtherNet/IP map transaction is incorrect.
Map Range Overruns	The requested transaction is reading or writing data to the internal EIP Table which is outside the boundaries of the table area.
Map Length Overruns	The size of the data received from the request is greater than the configured *Get Length* in the *EtherNet/IP Map*.

Table 7.8 - Parameters displayed in the Status Monitoring – EtherNet/IP Tab

Each of the explicit EtherNet/IP devices configured will also indicate if they are online or offline in the *Explicit EtherNet/IP Devices list* in the *EtherNet/IP* tab. The status can be either *Online* or *Offline*.

7.2.7. ETHERNET/IP ORIGINATOR

NOTE: The EtherNet/IP Originator tab is only displayed if the module has the primary interface set to EtherNet/IP Originator.

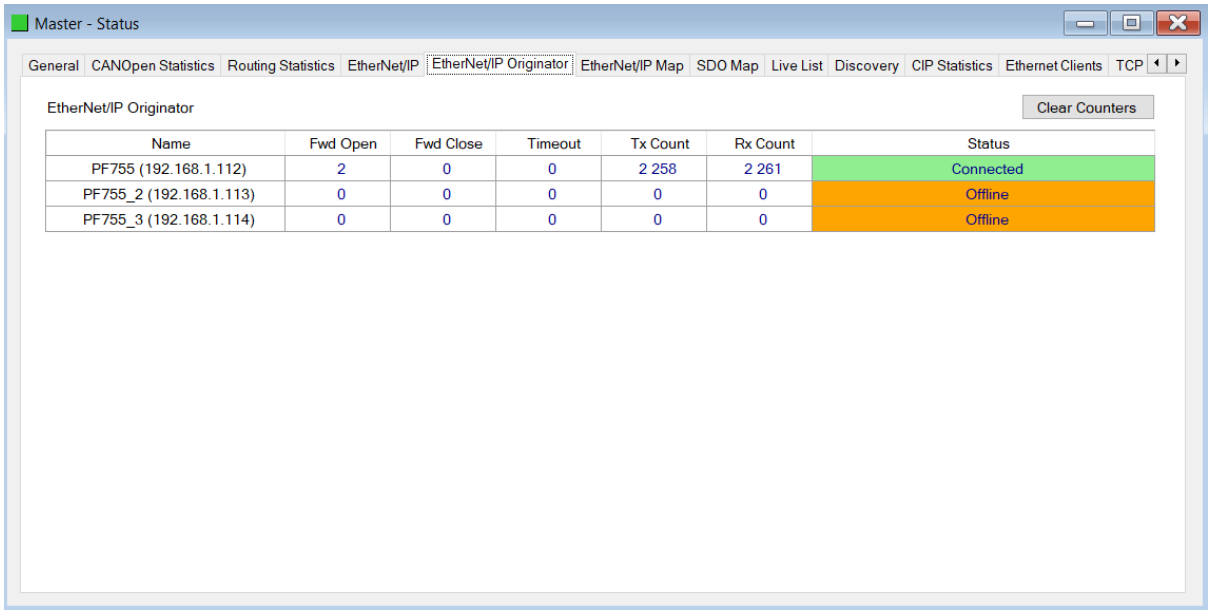


Figure 7.7. - Status monitoring – EtherNet/IP Originator

The EtherNet/IP tab displays the following parameters:

Parameter	Description
Name	The name of the configured connection.
Fwd Open	The number of Forward Open class 1 connection establishment requests sent.
Fwd Close	The number of Forward Close class 1 connection termination requests sent.
Timeout	The number of times the connected device has lost the class 1 connection due to not receiving any data within the allowed timeout time.
Tx Count	Number of class 1 cyclic packets sent to the target device.
Rx Count	Number of class 1 cyclic packets received from the target device.
Status	Status of the connection. **Connected** Class 1 connection has been established and is currently active. **Offline** Class 1 connection is **not** active.

Table 7.9 - Parameters displayed in the Status Monitoring – EtherNet/IP Tab

7.2.8. EtherNet/IP Map

NOTE: The EtherNet/IP Map tab is only displayed if the module has the primary interface set to EtherNet/IP Originator.

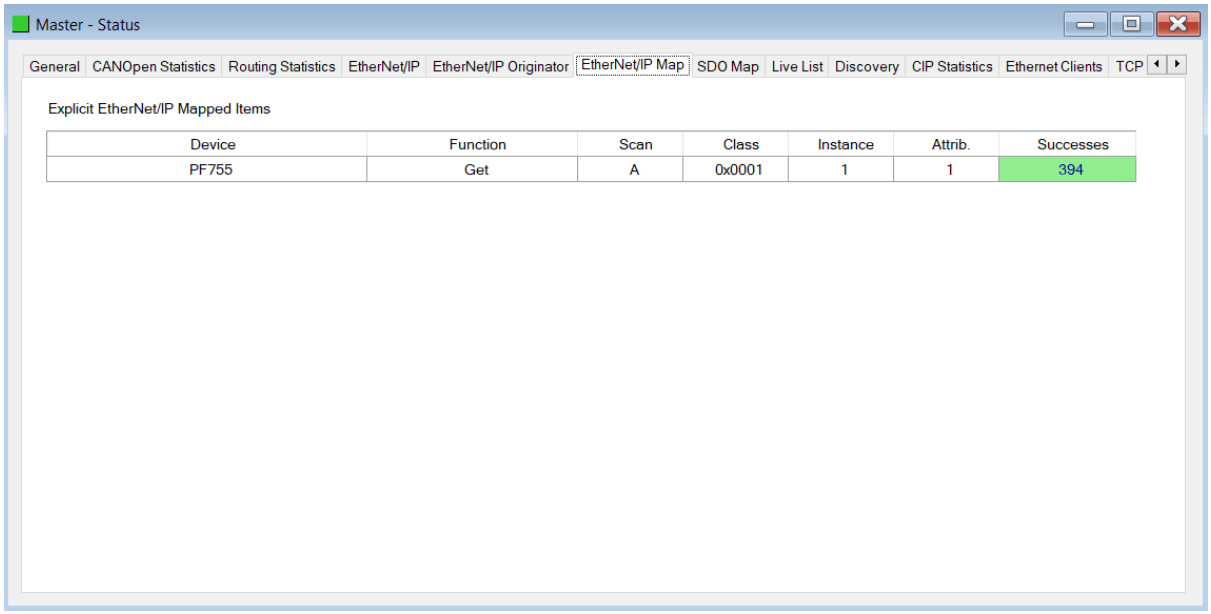


Figure 7.7. - Status monitoring – EtherNet/IP Map

The EtherNet/IP Map tab displays the following parameters:

Parameter	Description
Device	Name of the device configured in the EtherNet/IP Explicit devices.
Function	The function being used for the mapping item.
Scan	The scan class configured.
Class	The CIP Class for the request.
Instance	The CIP Instance for the request.
Attrib.	The CIP Attribute for the request.
Successes	The number successful transactions for this EtherNet/IP Explicit Mapped item.

Table 7.10 - Parameters displayed in the Status Monitoring – EtherNet/IP Map Tab

7.2.9. MODBUS STATISTICS

NOTE: The Modbus statistics tab is only displayed if the module has the primary interface set to Modbus Master or Modbus Slave.

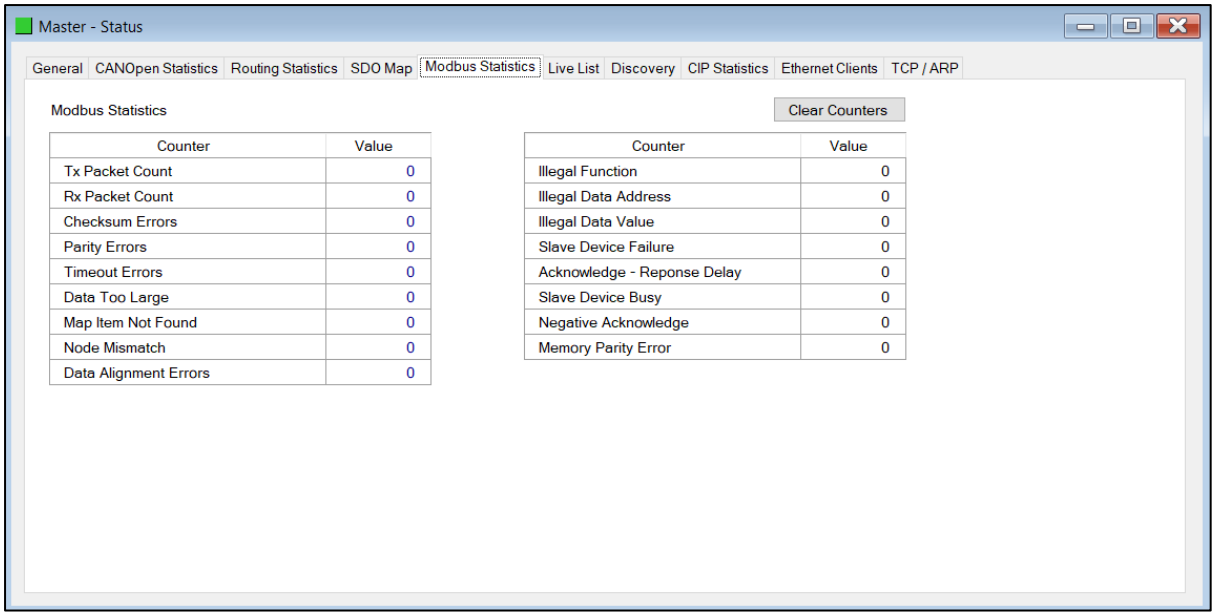


Figure 7.5. - Status monitoring – Modbus Statistics

The Modbus Statistics tab displays the following parameters:

Statistic	Description
Tx Packet Count	The number of Modbus packets sent by the module.
Rx Packet Count	The number of Modbus packets received by the module.
Checksum errors	The number of corrupted Modbus packets received by the module.
Parity errors	The number of bytes with parity errors received by the module.
Timeout Errors	The number of message response timeouts the module has encountered.
Data Too Large	The number of Modbus requests or responses where the data was too large to process.
Map Item Not Found	The number of Modbus requests did not match any mapped items.
Node Mismatch	The received Modbus request did not match the module's Modbus node address.
Data Alignment Errors	The Modbus request and associated mapped item is not byte aligned with the destination Logix tag.
Illegal Function	The number of times the Modbus device responded with an Illegal Function exception.
Illegal Data Address	The number of times the Modbus device responded with an Illegal Data Address exception.
Illegal Data Value	The number of times the Modbus device responded with an Illegal Data Value exception.
Slave Device Failure	The number of times the Modbus device responded with a Device Failure exception.

Acknowledge –Response Delay	The number of times the Modbus device responded with an Acknowledge exception.
Slave Device Busy	The number of times the Modbus device responded with a Slave Busy exception.
Negative Acknowledge	The number of times the Modbus device responded with a Negative Acknowledge exception.
Memory Parity Error	The number of times the Modbus device responded with a Memory Parity exception.

Table 7.11 - Parameters displayed in the Status Monitoring – Modbus Statistics Tab

7.2.10. SERIAL STATISTICS

The Serial Statistics tab displays the Serial statistics for the Modbus Read and Write Message Exchanges when the module is a Modbus Master or Modbus Slave.

NOTE: The Serial statistics tab is only displayed if the module has the primary interface set to Modbus Master or Modbus Slave.

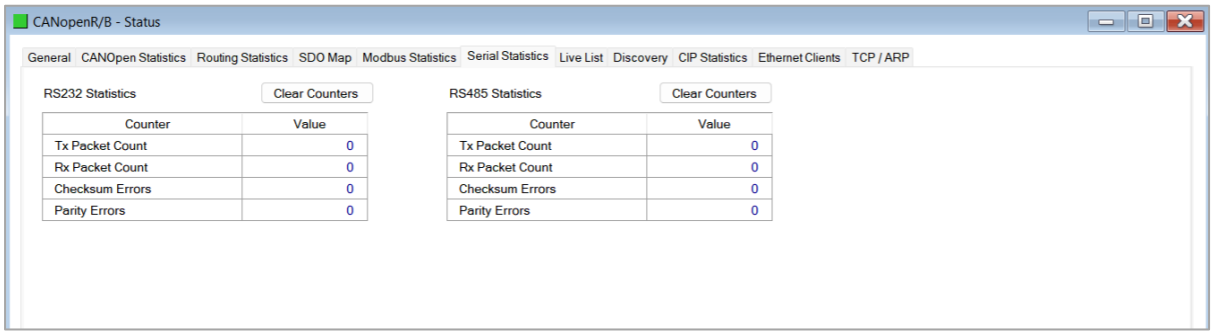


Figure 7.6. - Status monitoring – Serial Statistics

The Serial Statistics tab displays the following parameters:

Statistic	Description
RS232	
Tx Packet Count	The number of RS232 packets sent by the module.
Rx Packet Count	The number of RS232 packets received by the module.
Checksum errors	The number of corrupted RS232 packets received by the module.
Parity errors	The number of RS232 bytes with parity errors received by the module.
RS485	
Tx Packet Count	The number of RS485 packets sent by the module.

Rx Packet Count	The number of RS485 packets received by the module.
Checksum errors	The number of corrupted RS485 packets received by the module.
Parity errors	The number of RS485 bytes with parity errors received by the module.

Table 7.12 - Serial Statistics Tab

7.2.11. SDO MAP

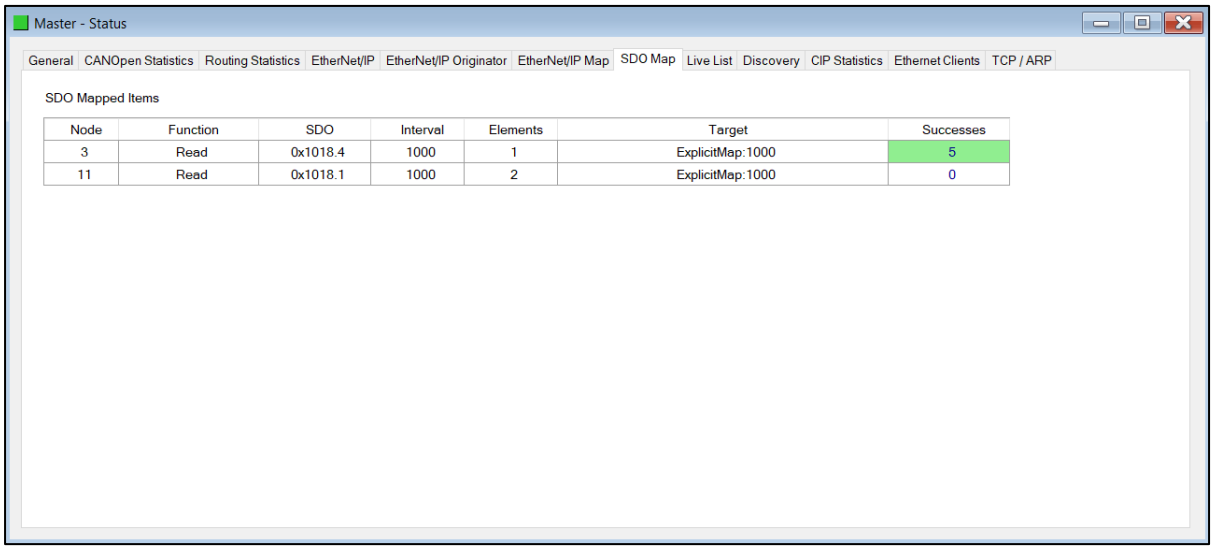


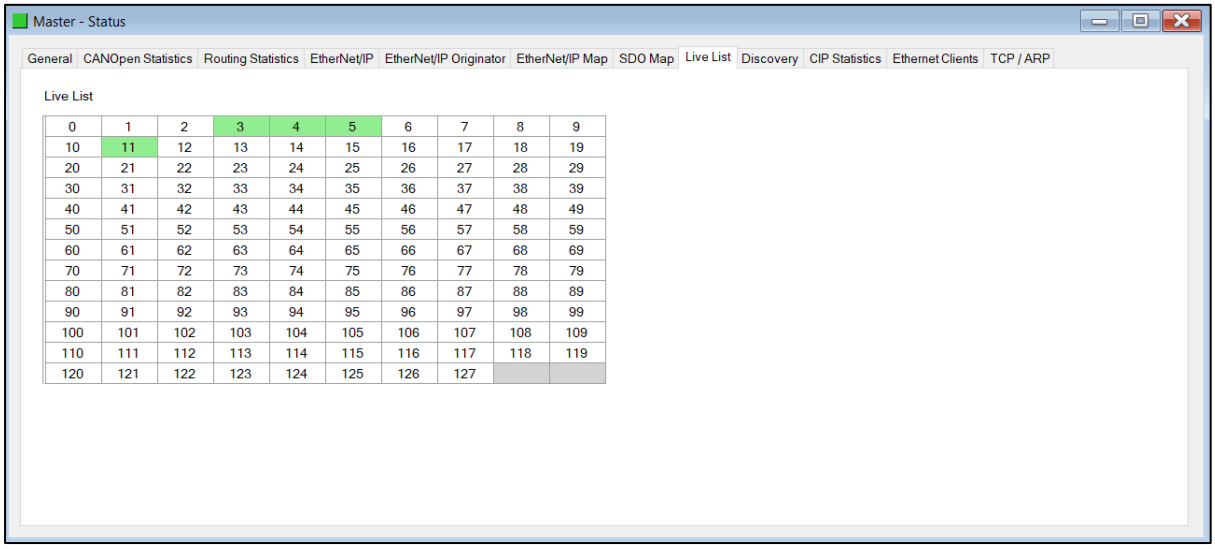
Figure 7.7. - Status monitoring – SDO Map

The SDO Map tab displays the following parameters:

Statistic	**Description**
Node	The configured node number of the SDO map item.
Function	The configured function of the SDO map item. This will be either read or write.
SDO	The SDO parameter index and sub-index configured for the SDO map item.
Interval	The interval (in milliseconds) that the transaction will be executed.
Elements	The number of elements configured for the SDO map item.
Target	The configured target destination of the transaction. This will differ for each primary interface.
Successes	The number successful transactions for this SDO Map item.

Table 7.13 - Parameters displayed in the Status Monitoring – SDO Map Tab

7.2.12. LIVE LIST

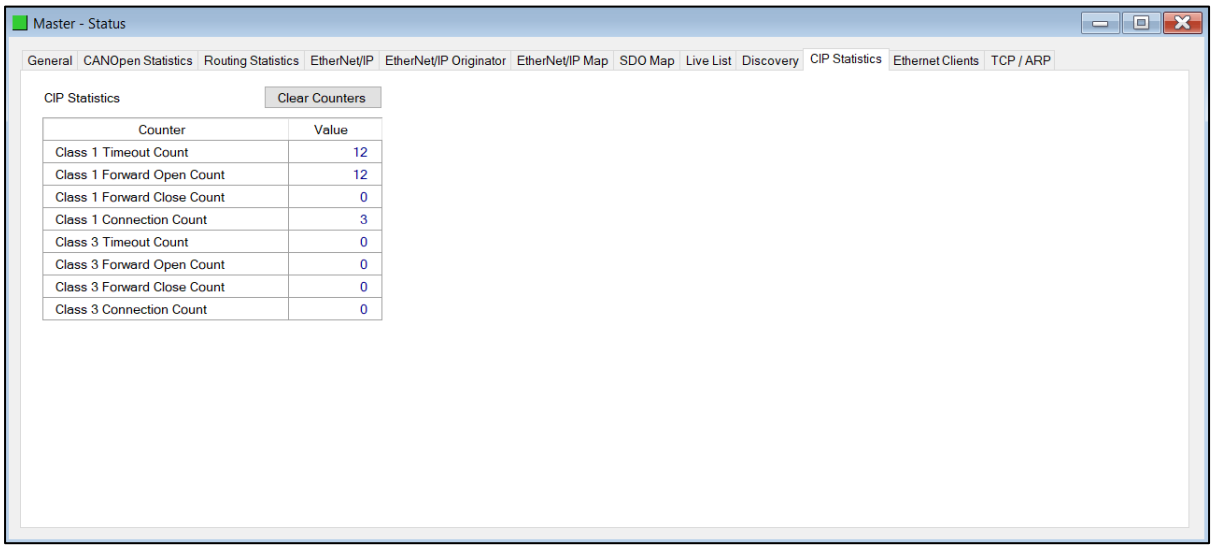


0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109
110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127		

Figure 7.8. - Status monitoring – Live List

The Live List provides the online status of each CANopen Slave device on the CANopen network. The address in the live list is highlighted in green when the last response received from the slave is less than the *Slave Inactive Timeout* parameter in the CAN Bus configuration (i.e. the device is considered to be online).

7.2.13. CIP STATISTICS



Counter	Value
Class 1 Timeout Count	12
Class 1 Forward Open Count	12
Class 1 Forward Close Count	0
Class 1 Connection Count	3
Class 3 Timeout Count	0
Class 3 Forward Open Count	0
Class 3 Forward Close Count	0
Class 3 Connection Count	0

Figure 7.9. - Status monitoring – CIP statistics

The CIP statistics tab displays the following parameters:

Statistic	Description
Class 1 Timeout Count	Number of times a Class 1 connection has timed out
Class 1 Forward Open Count	Number of Class 1 Connection establish attempts
Class 1 Forward Close Count	Number of Class 1 Connection close attempts
Class 1 Connection Count	Number of Class 1 Connections currently active
Class 3 Timeout Count	Number of times a Class 3 connection has timed out
Class 3 Forward Open Count	Number of Class 3 Connection establish attempts
Class 3 Forward Close Count	Number of Class 3 Connection close attempts
Class 3 Connection Count	Number of Class 3 Connections currently active

Table 7.14. - CIP Statistics

7.2.14. ETHERNET CLIENTS

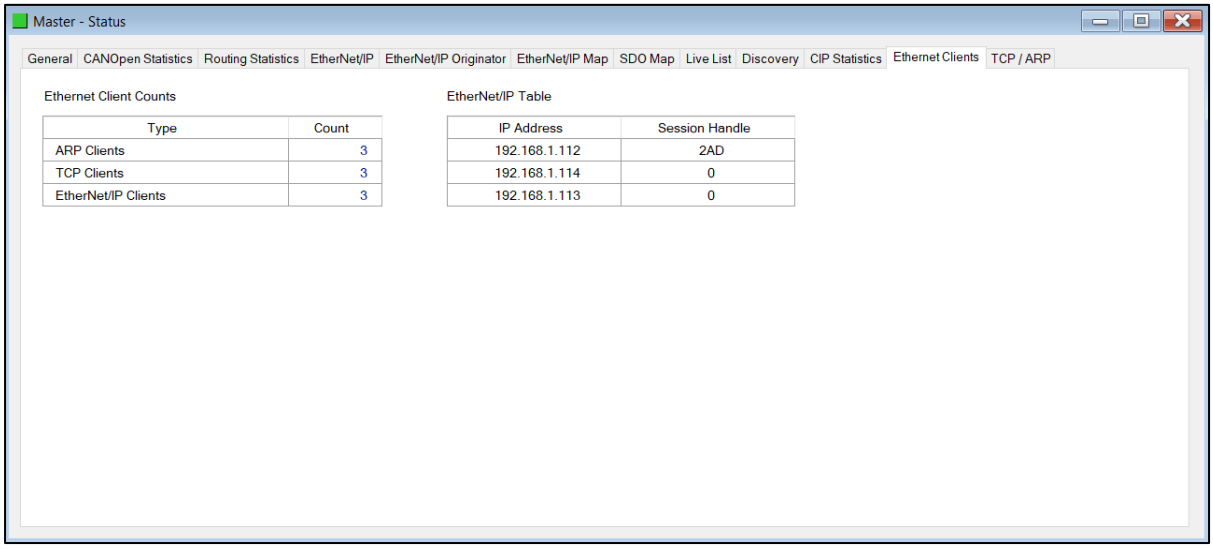


Figure 7.10. - Status monitoring – Ethernet Clients

The Ethernet Clients tab displays the following parameters:

Statistic	Description
Ethernet Clients	
ARP Clients	The number of ARP (Address Resolution Protocol) Clients
TCP Clients	The number of TCP (Transmission Control Protocol) Clients
EtherNet/IP Clients	The number of EtherNet/IP Clients
EtherNet/IP Table	

IP Address	IP Address of the remote EtherNet/IP client
Session Handle	Session Handle associated with the EtherNet/IP connection

Table 7.15. – Ethernet Clients

7.2.15. TCP / ARP

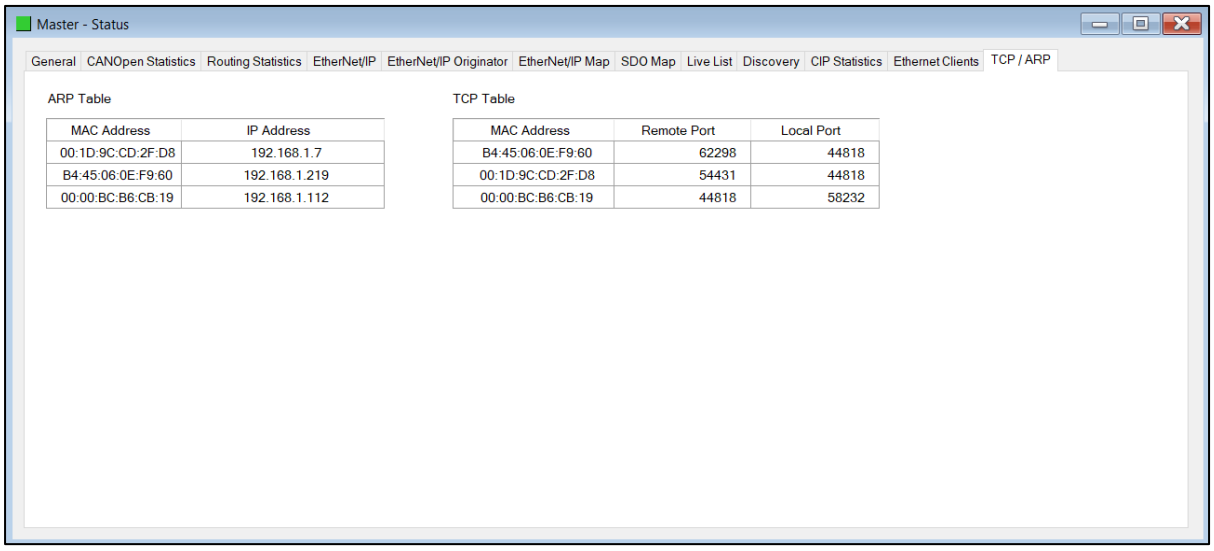


Figure 7.11. - Status monitoring – TCP / ARP

The TCP / ARP tab displays the following parameters:

Statistic	Description
ARP Table	
MAC Address	The MAC address of the remote Ethernet interface
IP Address	The IP (Internet Protocol) address
TCP Table	
MAC Address	The MAC address of the remote Ethernet interface
Remote Port	The TCP Port of the remote device
Local Port	The TCP port of the local device

Table 7.16. – TCP / ARP

7.3. SLAVE DEVICE STATUS MONITORING IN SLATE

To view the CANopen Slave device's status in the Aparian-Slate environment, the module must be online. If the module is not already Online (following a recent configuration download), then right-click on the module and select the *Go Online* option.

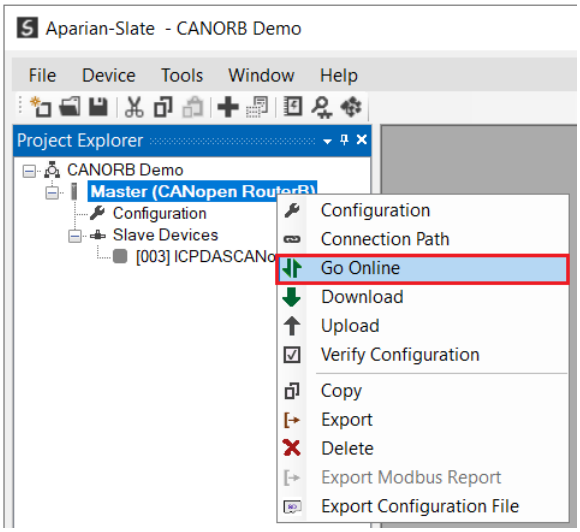


Figure 7.12. - Selecting to Go Online

The Online mode is indicated by the green circle behind the module in the Project Explorer tree.

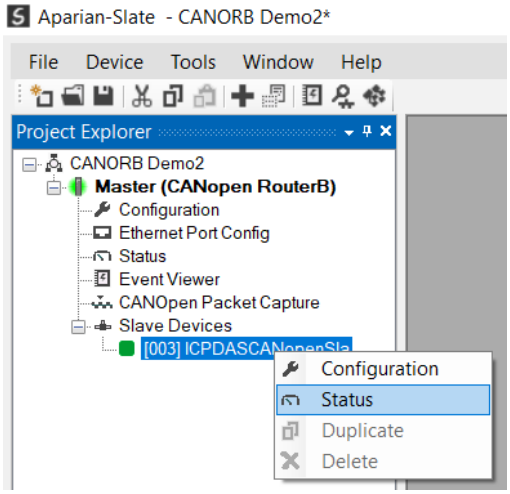


Figure 7.13. - Selecting Slave Device Online Status

The Status monitoring window can be opened by right-clicking on the CANopen Slave device and selecting *Status*.

The status window contains multiple tabs to display the status of the CANopen Slave device.

7.3.1. GENERAL

Most of these parameters in the status windows are self-explanatory or have been discussed in previous sections.

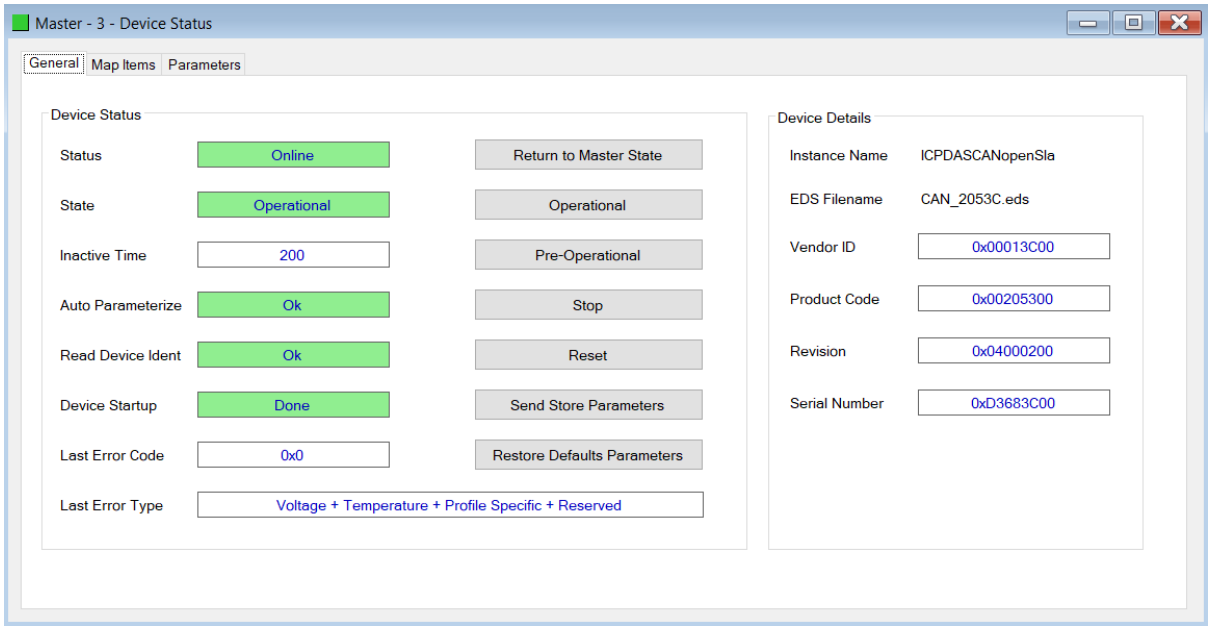


Figure 7.14. – CANopen Slave Device Online Status

The CANopen Slave Device Status tab displays the following general parameters:

Statistic	Description
Device Status	
Status	The current online/offline status of the slave device.
State	The current state of the slave device. *Operational* *Pre-operational* *Stopped*
Inactive Time	The amount of time (in milliseconds) that have elapsed since the last response from the slave device.
Auto Parameterize	Status of the Auto parameterize of the CANopen Slave device during startup.
Read Device Ident	Status of the Device Ident reading to indicate if it was successful or failed.
Device Startup	Indication of the device status during startup. This can be either *Done* or *Busy* informing the user the device is still being read or setup.
Last Error Code	The last error code received from the slave device.
Last Error Type	The last error type received from the slave device.

Device Details	
Instance Name	The instance name configured for the module in the device configuration.
EDS Filename	The EDS file being used.
Vendor ID	The Vendor ID read from the device during startup.
Product Code	The Product Code read from the device during startup.
Revision	The Revision read from the device during startup.
Serial Number	The Serial Number read from the device during startup.

Table 7.17. - CANopen Slave Device Online Status

7.3.2. MAP ITEMS

The map items tab shows the status of each PDO. The item will go green every time there is a successful transaction. Timeouts will occur when data has not been received at a configured interval (e.g. when using Remote-TxReq transmission type).

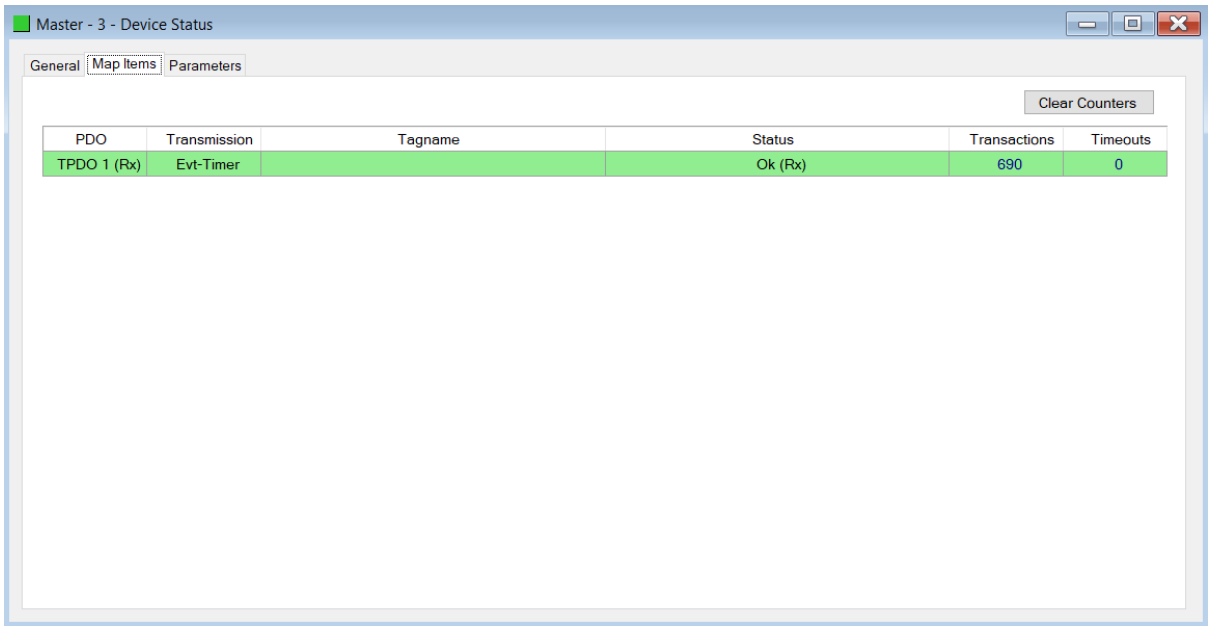


Figure 7.15. – CANopen Slave Device Status - Map Items

7.4. CANOPEN PACKET CAPTURE

The module provides the capability to capture the CANopen traffic for analysis. The will allow the user and a remote support team to resolve any possible issues on site. To invoke the

capture of the module, double-click on the CANopen Packet Capture item in the Project Explorer tree.

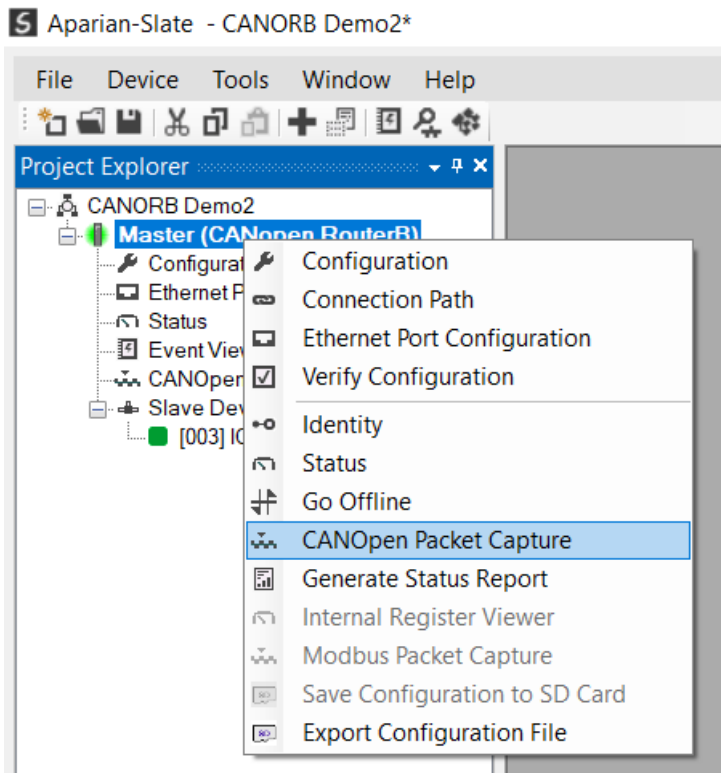


Figure 5.16 - Selecting CANopen Packet Capture

The CANopen Packet Capture window will open and automatically start capturing all CANopen packets.

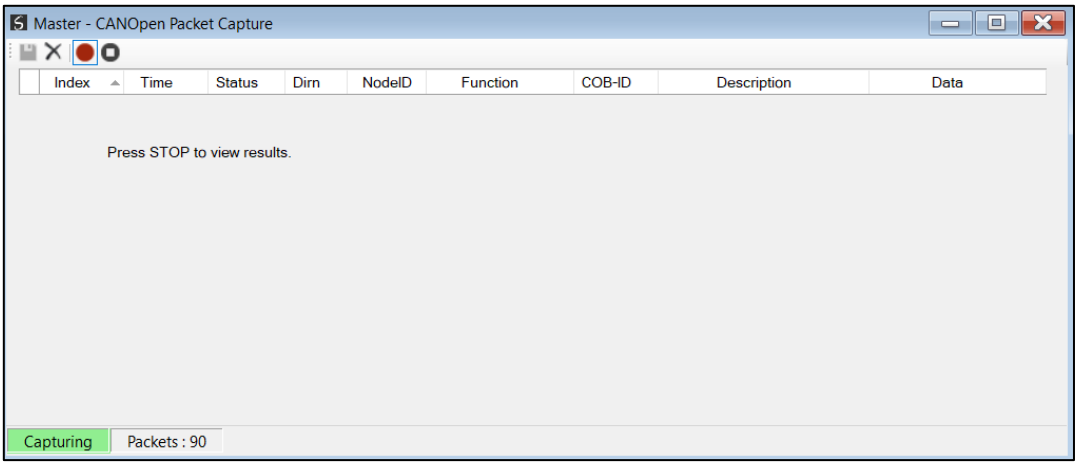


Figure 5.17 – CANopen packet capture

To display the captured CANopen packets, the capture process must first be stopped, by pressing the Stop button.

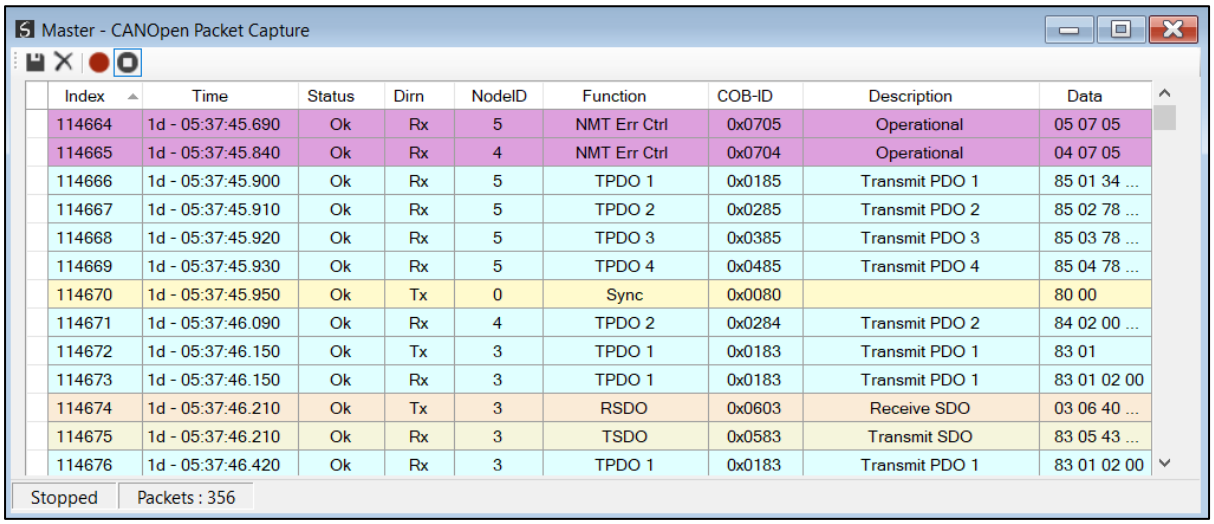


Figure 5.18 – CANopen Packet Capture complete

The captured CANopen packets are tabulated as follows:

Statistic	Description
Index	The packet index, incremented for each packet sent or received.
Time	The elapsed time since the module powered up.
Status	The status of the packet. Received packets are checked for valid CANopen constructs and valid checksums.
Dirn	The direction of the packet, either transmitted (Tx) or received (Rx).
NodeID	The Source Node address for the packet
Function	The CANopen function.
COB-ID	The COB-ID for the specific packet.
Description	Description of the packet that was received.
Data	The raw packet data.

Table 5.18 – CANopen Packet Capture fields

The packet capture can be saved to a file for further analysis, by selecting the ***Save*** button on the toolbar. Previously saved CANopen Packet Capture files can be viewed by selecting the ***CANopen Packet Capture Viewer*** option in the tools menu.

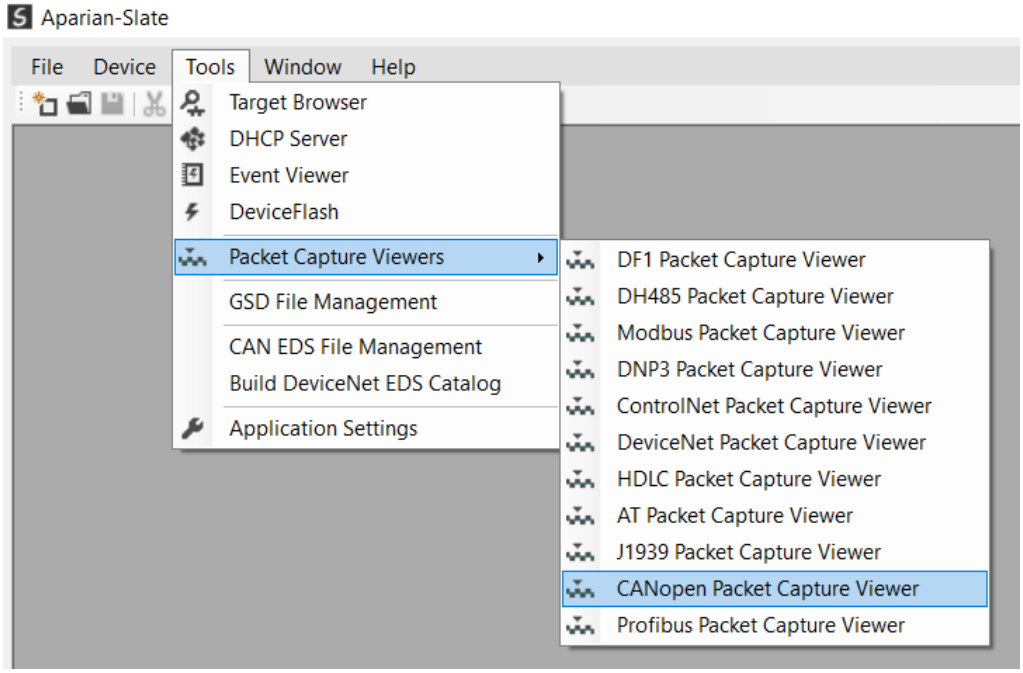


Figure 5.19 - Selecting the CANopen Packet Capture Viewer

7.5. MODBUS PACKET CAPTURE

The module provides the capability to capture the Modbus traffic for analysis. The will allow the user and a remote support team to resolve any possible issues on site. To invoke the capture of the module, double-click on the Modbus Packet Capture item in the Project Explorer tree.

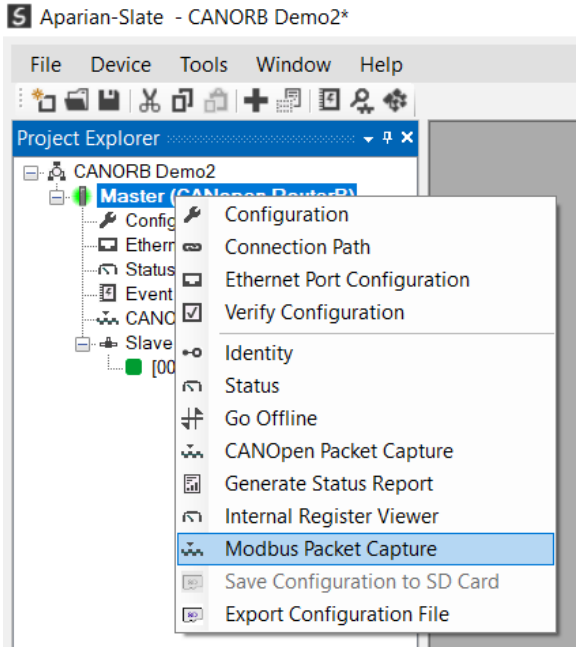


Figure 5.20 - Selecting Modbus Packet Capture

The Modbus Packet Capture window will open and automatically start capturing all Modbus packets.

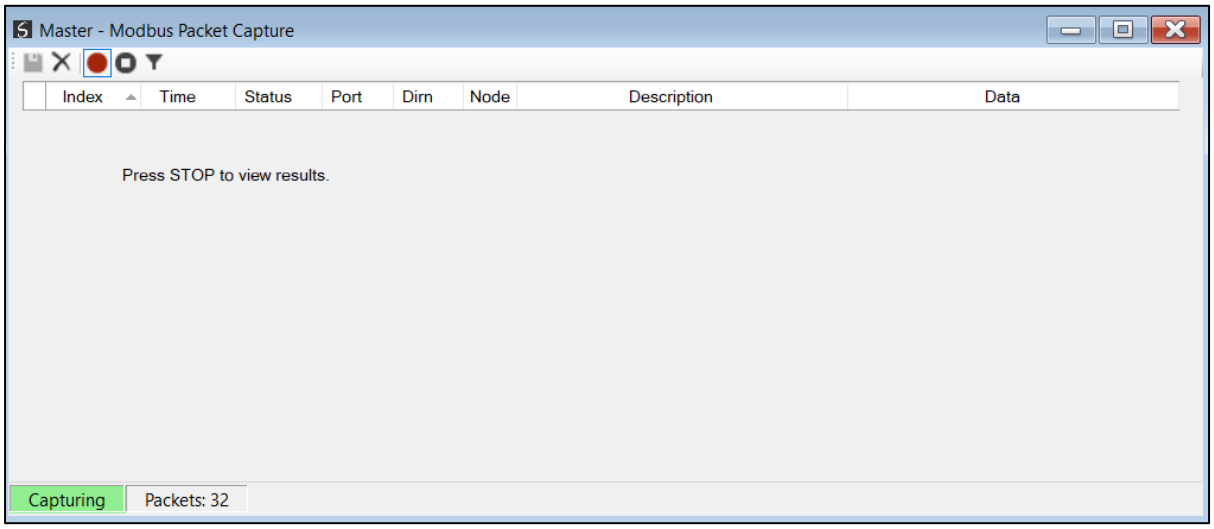


Figure 5.21 – Modbus packet capture

To display the captured Modbus packets, the capture process must first be stopped, by pressing the Stop button.

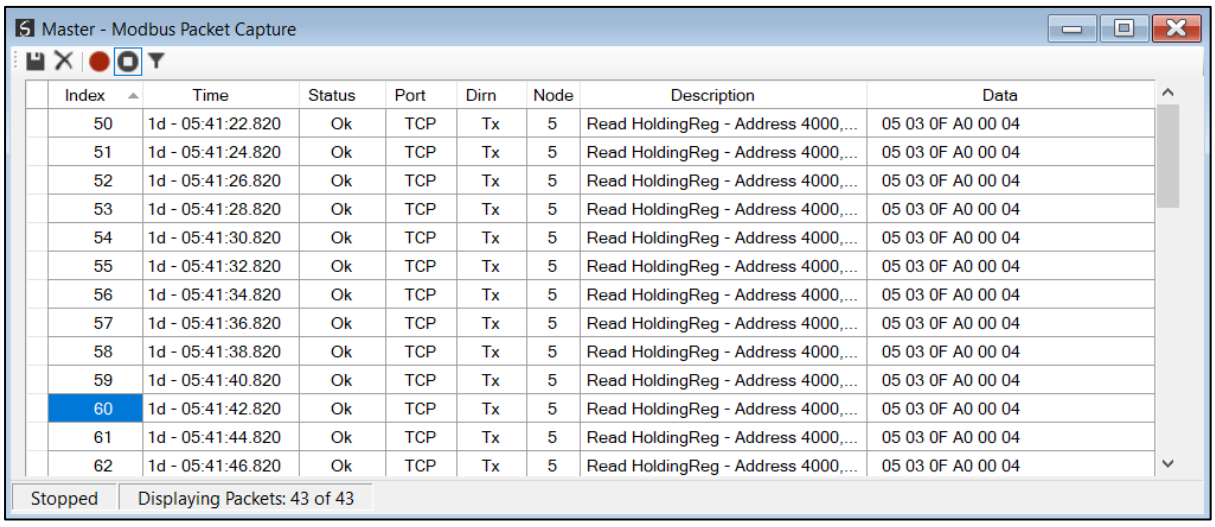


Figure 5.22 – Modbus Packet Capture complete

The captured Modbus packets are tabulated as follows:

Statistic	Description
Index	The packet index, incremented for each packet sent or received.

Time	The elapsed time since the module powered up.
Status	The status of the packet. Received packets are checked for valid Modbus constructs and valid checksums.
Port	Port on where the data was sent or received (TCP, RTU232, RTU485)
Dirn	The direction of the packet, either transmitted (Tx) or received (Rx).
Node	The Source Node address for the packet
Description	Description of the packet that was received.
Data	The raw packet data.

Table 5.19 – Modbus Packet Capture fields

The packet capture can be saved to a file for further analysis, by selecting the ***Save*** button on the toolbar. Previously saved Modbus Packet Capture files can be viewed by selecting the ***Modbus Packet Capture Viewer*** option in the tools menu.

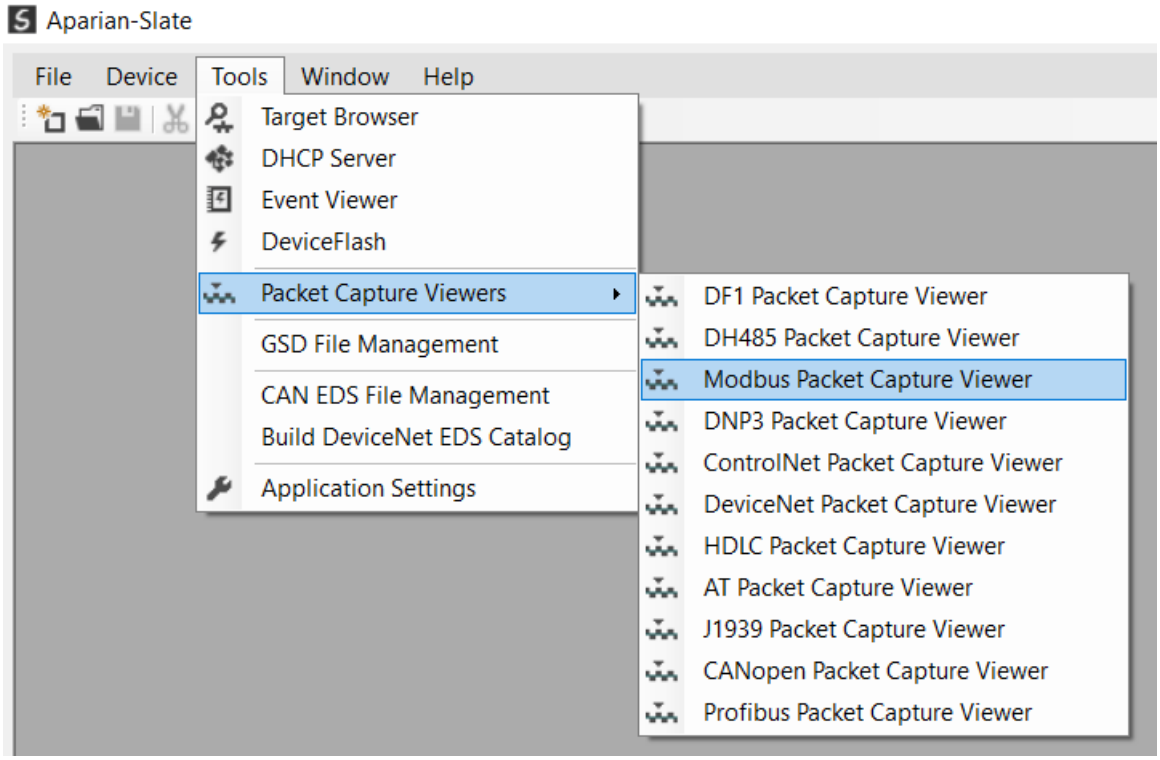


Figure 5.23 - Selecting the Modbus Packet Capture Viewer

7.6. MODULE EVENT LOG

The CANopen Router/B module logs various diagnostic records to an internal event log. These logs are stored in non-volatile memory and can be displayed using Slate or via the web interface. To view them in Slate, select the ***Event Viewer*** option in the Project Explorer tree.

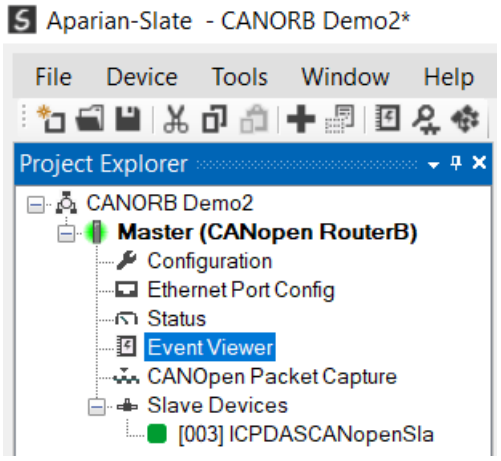
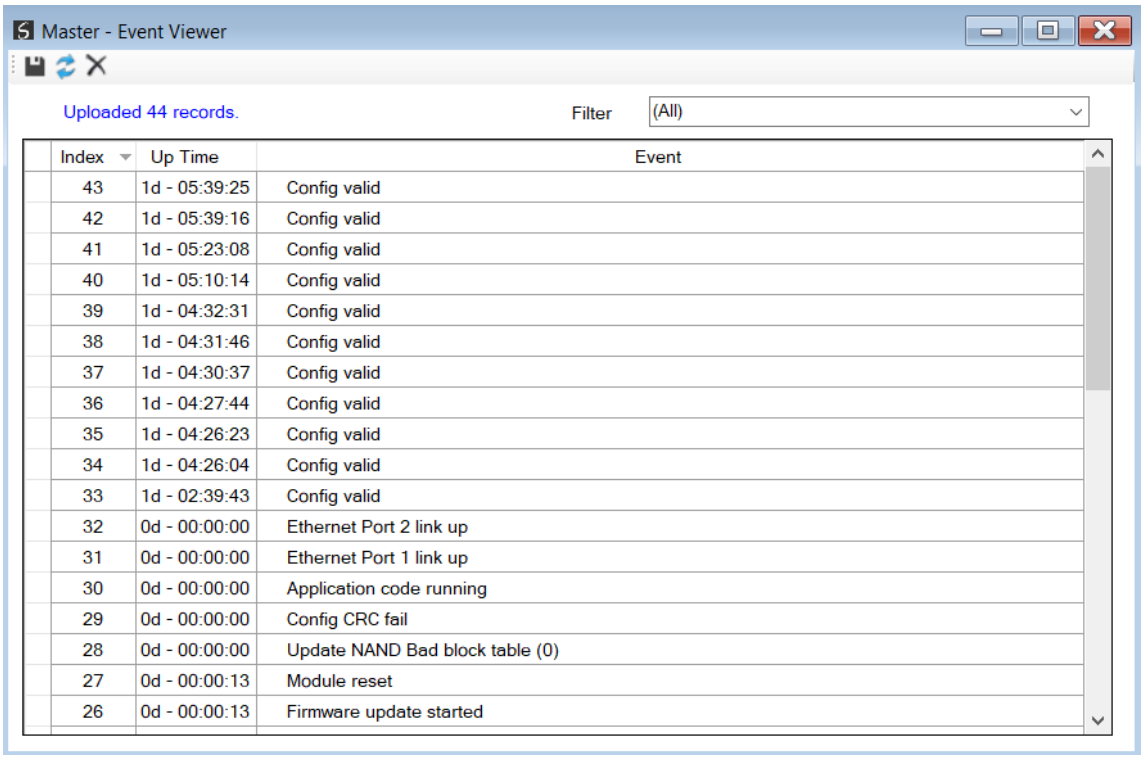


Figure 7.24. - Selecting the module Event Log

The Event Log window will open and automatically read all the events from the module. The log entries are sorted so as to have the latest record at the top. Custom sorting is achieved by double-clicking on the column headings.



Index	Up Time	Event
43	1d - 05:39:25	Config valid
42	1d - 05:39:16	Config valid
41	1d - 05:23:08	Config valid
40	1d - 05:10:14	Config valid
39	1d - 04:32:31	Config valid
38	1d - 04:31:46	Config valid
37	1d - 04:30:37	Config valid
36	1d - 04:27:44	Config valid
35	1d - 04:26:23	Config valid
34	1d - 04:26:04	Config valid
33	1d - 02:39:43	Config valid
32	0d - 00:00:00	Ethernet Port 2 link up
31	0d - 00:00:00	Ethernet Port 1 link up
30	0d - 00:00:00	Application code running
29	0d - 00:00:00	Config CRC fail
28	0d - 00:00:00	Update NAND Bad block table (0)
27	0d - 00:00:13	Module reset
26	0d - 00:00:13	Firmware update started

Figure 7.25. – Module Event Log

The log can also be stored to a file for future analysis, by selecting the ***Save*** button in the tool menu. To view previously saved files, use the ***Event Log Viewer*** option under the ***Tools*** menu.

7.7. WEB SERVER

The CANopen Router/B provides a web server allowing a user without Slate or Logix 5000 to view various diagnostics of the module. This includes Ethernet parameters, system event log, advanced diagnostics, and application diagnostics (e.g. CANopen statistics).

NOTE: The web server is view **only** and thus no parameters or configuration can be altered from the web interface.

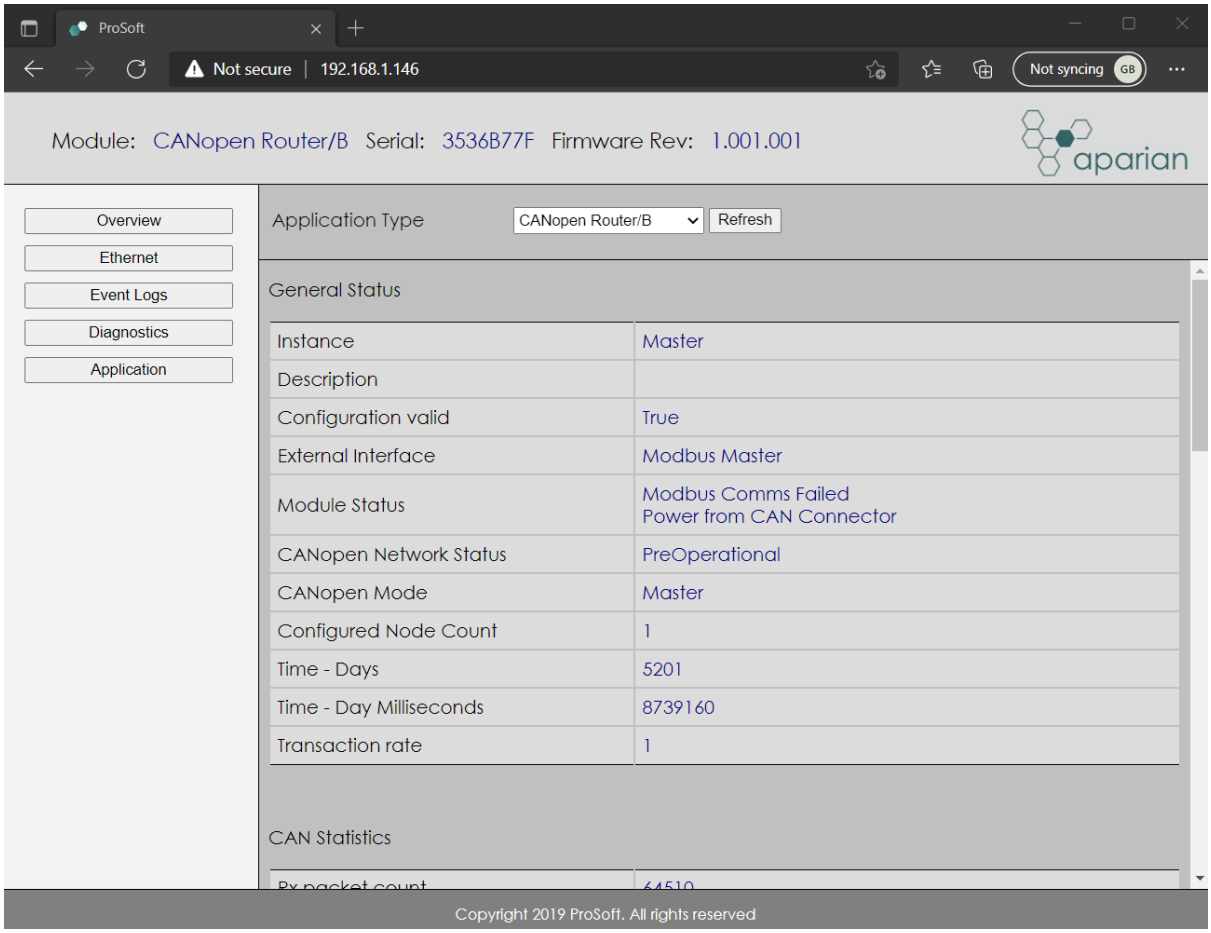


Figure 7.26. - Web interface

7.8. MODULE STATUS REPORT

For assisting with support Slate can generate a status report for the module which is a word document that can be emailed to support. To generate this report the user can right-click on the module (when online in Slate) and select *Generate Status Report*.

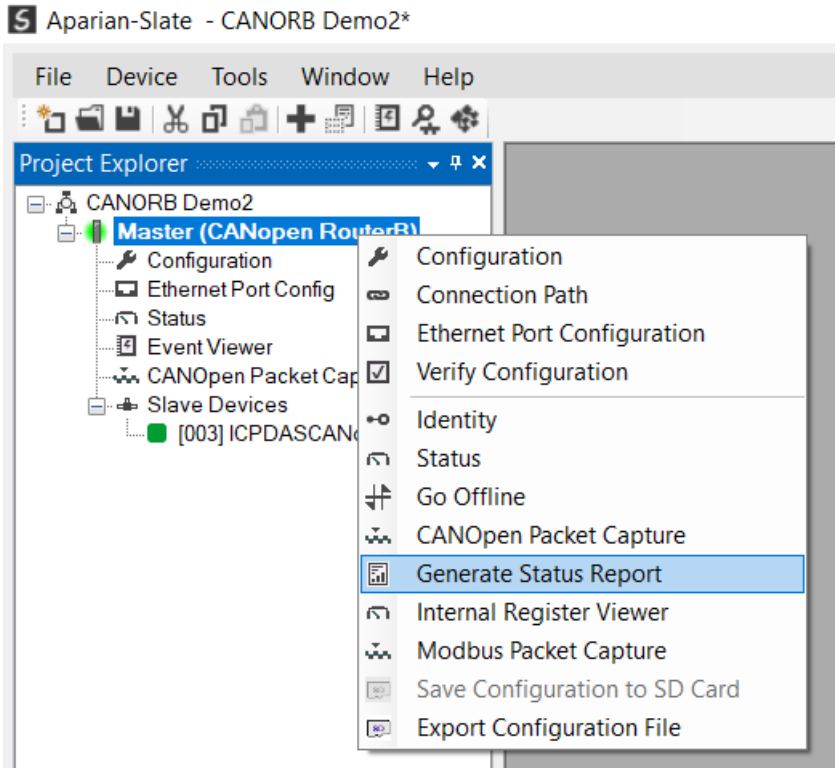


Figure 7.27. – Module Status Report

7.9. INTERNAL REGISTER VIEWER

The module provides an Internal Register Viewer that allows the user to view the data in the Modbus Registers.

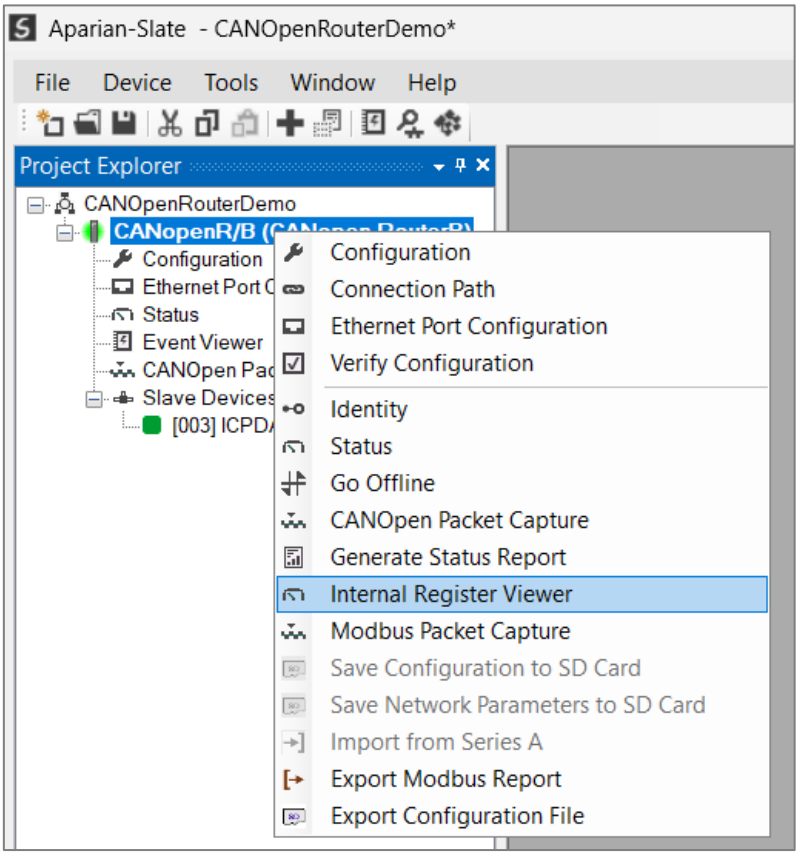


Figure 7.28 – Internal Register Viewer - Select

Once opened, the register type can be selected. Note that the Modbus Registers will only be available if the Primary Interface configured is for Modbus Master or Modbus Slave.

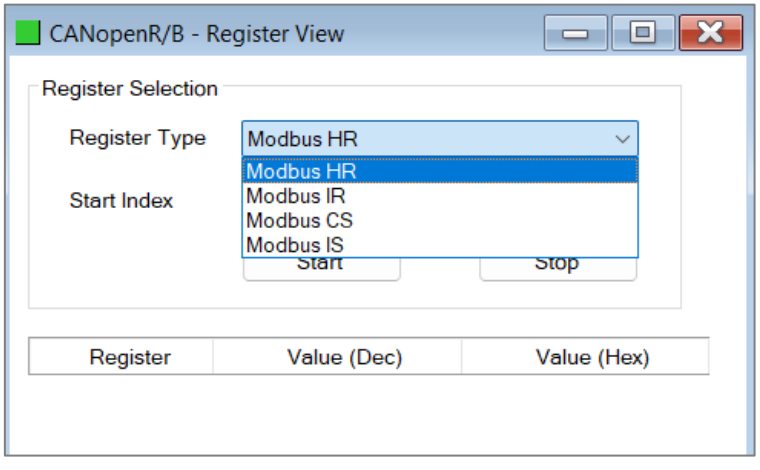


Figure 7.29 – Internal Register Viewer – Register Type

The Start and Stop Index can then be selected to determine how much data to display and refresh.

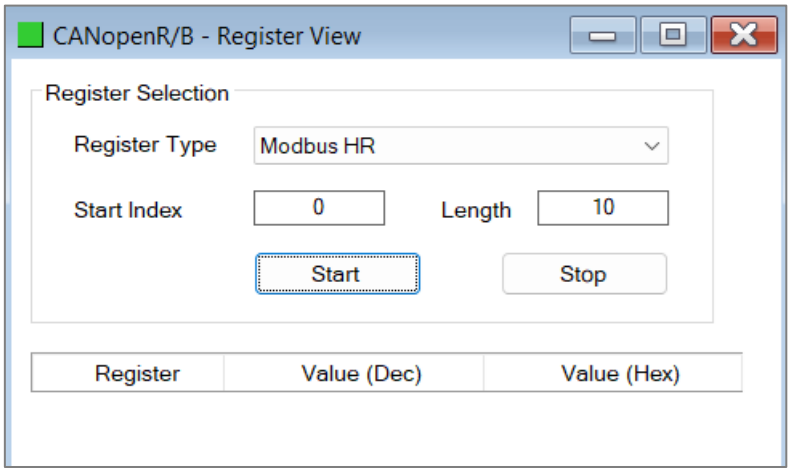


Figure 7.30 – Internal Register Viewer – Indexes

The data will appear and start refreshing by pressing the *Start* button.

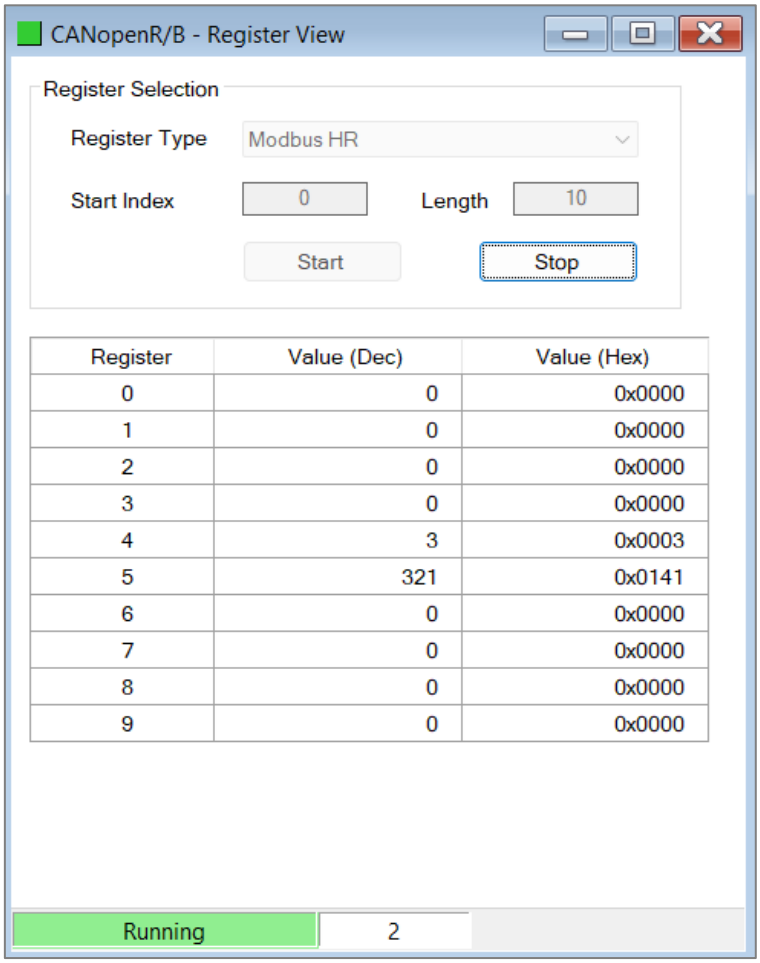


Figure 7.31 – Internal Register Viewer – Running

8. TECHNICAL SPECIFICATIONS

8.1. DIMENSIONS

Below are the enclosure dimensions as well as the required DIN rail dimensions. All dimensions are in millimetres.

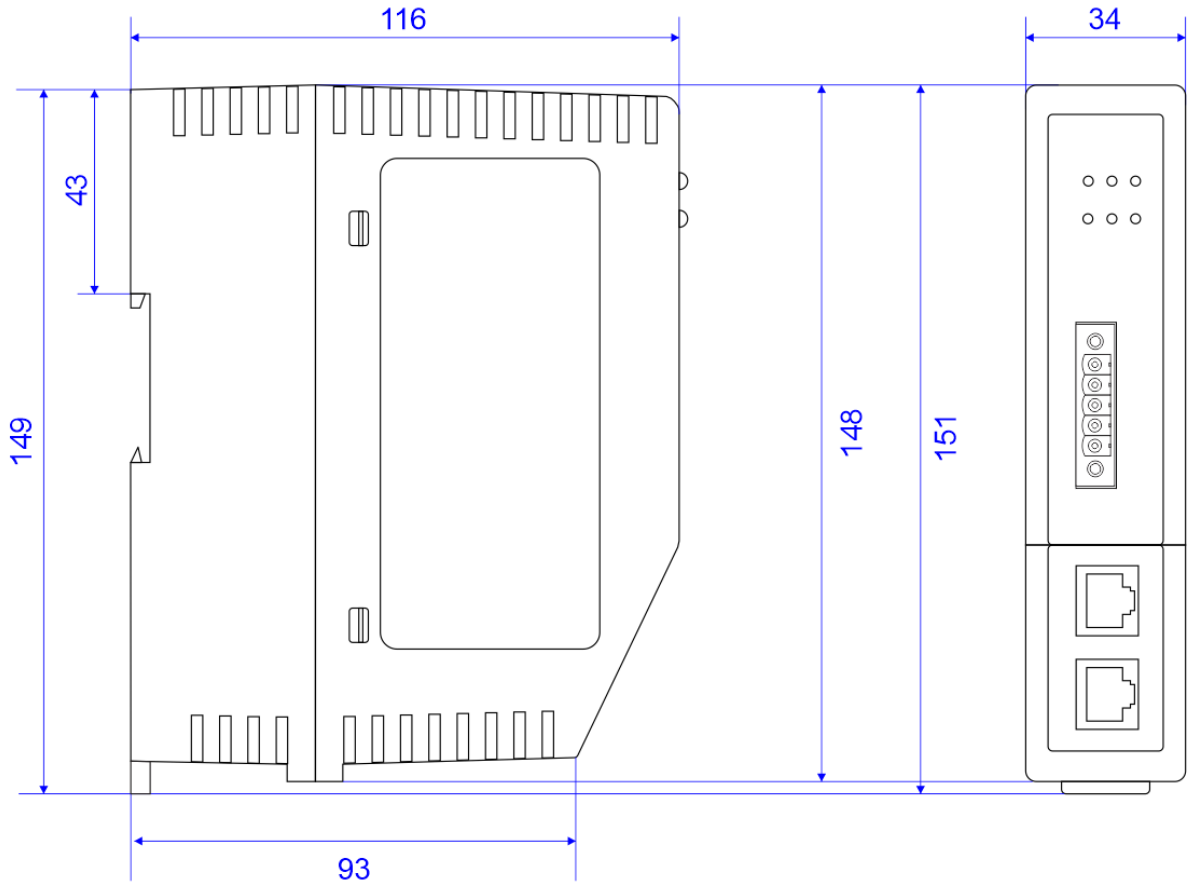


Figure 8.1 – CANopen Router/B enclosure dimensions

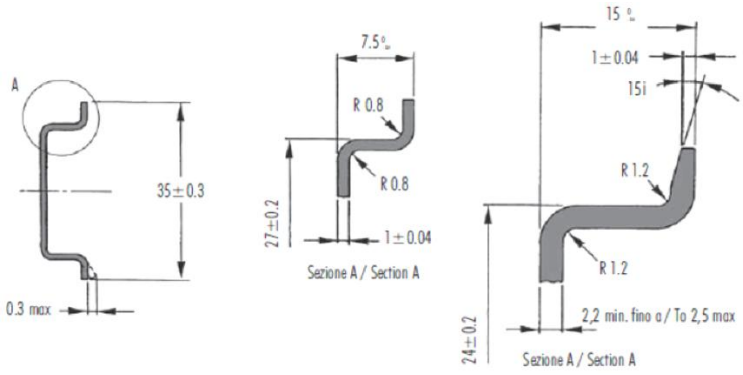


Figure 8.2 - Required DIN dimensions

8.2. ELECTRICAL

Specification	Rating
Power requirements	Input: 10 – 32V DC
Power consumption	2.2 W (Max.) Current: 180 mA @ 10 V Current: 85 mA @ 24 V
Connector	3-way terminal
Conductors	24 – 18 AWG
Enclosure rating	IP20, NEMA/UL Open Type
Temperature	-20 – 70 °C
Earth connection	Yes, terminal based
Emissions	IEC61000-6-4
ESD Immunity	EN 61000-4-2
Radiated RF Immunity	IEC 61000-4-3
EFT/B Immunity	EFT: IEC 61000-4-4
Surge Immunity	Surge: IEC 61000-4-5
Conducted RF Immunity	IEC 61000-4-6

Table 8.1 - Electrical specification

8.3. ETHERNET

Specification	Rating
Connector	RJ45
Conductors	CAT5 STP/UTP
ARP connections	Max 200
TCP connections	Max 200
CIP connections	Max 15
Communication rate	10/100Mbps
Duplex mode	Full/Half
Auto-MDIX support	Yes

Embedded switch	Yes, 2 x Ethernet ports
Device Level Ring (DLR)	Supported
Network Time Protocol (NTP)	Supported

Table 8.2 - Ethernet specification

8.4. SERIAL PORT (RS232)

Specification	Rating
RS232 Connector	9-way terminal (shared with RS485)
RS232 Conductor	24 – 18 AWG
Electrical Isolation	1000 Vdc
BAUD	1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
Parity	None, Even, Odd
Data bits	8
Stop bits	1

Table 8.3 – RS232 Serial Port specification

8.5. SERIAL PORT (RS485)

Specification	Rating
RS485 Connector	9-way terminal (shared with RS485)
RS485 Conductor	24 – 18 AWG
Electrical Isolation	1500 Vrms for 1 minute.
BAUD	1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
Parity	None, Even, Odd
Data bits	8
Stop bits	1

Table 8.4 – RS485 Serial Port specification

8.6. CANOPEN

Specification	Rating
Connector	5-way terminal, 5.08mm pitch.
Modes	CANopen Master CANopen Slave
Supported Baud Rates	10k 20k 50k 125k 250k 500k 800k 1M
CANopen Terminator	120 Ω - Software Enabled

Table 8.5 – CANopen specification

NOTE: Although the CANopen Router supports the CiA443 Auto-enable objects, the CANopen interface is not fault-tolerant.

8.7. CANOPEN MASTER

Specification	Rating
CANopen Slave Count	124
PDO Count per Device	32
SDO mapping count	128
MPDO supported	Yes
CANopen Slave Auto Parameterize	Supported
CiA 443 Bootloader auto-enable Support	Yes
NMT messages	Operational Control (e.g. Stopped, Pre-operational, Operational) SYNC TIME EMCY
Layer Setting Services (LSS)	Node and BitRate assignment supported

Implementation	CiA 301 v4.2.0

Table 8.6 – CANopen Master specification

8.8. CANOPEN SLAVE

Specification	Rating
CANopen PDO emulation count	125
CANopen Emulated devices supported	125
MPDO supported	Yes

Table 8.7 – CANopen Slave specification

8.9. ETHERNET/IP TARGET

Specification	Rating
Class 1 Cyclic connection count	4
Logix Direct-to-Tag Supported	Yes

Table 8.8 – EtherNet/IP Target specification

8.10. ETHERNET/IP ORIGINATOR

Specification	Rating
Class 1 Cyclic Connections Supported	Yes
Class 3 / UCMM Connections Supported	Yes
Class 1 Connection Count	5
Class 3 / UCMM Target Device Count	5
Class 3 / UCMM Mapping Count	50

Table 8.9 – EtherNet/IP Originator specification

8.11. MODBUS MASTER

Specification	Rating
Modes Supported	Modbus TCP, Modbus RTU232, Modbus RTU485
Modbus RTU485 Termination	125 Ω - Software Enabled
Max Modbus Slave device	20
Max Modbus Mapping	100
Mapping Ranges	Holding Register 0 – 65535 Input Register 0 – 65535 Input Status 0 – 65535 Coil Status 0 – 65535
Base Offset	Modbus (Base 0) PLC (Base 1)
Configurable Modbus TCP Port	Yes
Data Reformatting Supported	BB AA BB AA DD CC CC DD AA BB DD CC BB AA

Table 8.10 – Modbus Master specification

8.12. MODBUS SLAVE

Specification	Rating
Modes Supported	Modbus TCP, Modbus RTU232, Modbus RTU485 (simultaneous)
Modbus RTU485 Termination	Software set
Mapping Ranges	Holding Register 0 – 65535 Input Register 0 – 65535 Input Status 0 – 65535 Coil Status 0 – 65535
Base Offset	Modbus (Base 0) PLC (Base 1)
Configurable Modbus TCP Port	Yes

Table 8.11 – Modbus Slave specification

8.13. CERTIFICATIONS

Certification	Mark
CE Mark	CE
RoHS2 Compliant	RoHS2
UL Mark File: E494895	c UL us LISTED CLASS 1, DIV 2, GROUPS A, B, C, D
ODVA Conformance	EtherNet/IP™
ATEX	Ex II 3 G Ex ec IIC T5 -25°C ≤ Ta ≤ 70 °C
UKCA	UK CA

Table 8.12 – Certifications

9. INDEX