CANopen Router

User Manual

A-CANOR

Document No. D124-007
01/2020
Revision 1.3

aparian

CONTENTS

Revision History

Revision	Date	Comment
1.0	7 December 2018	Initial document
1.1	25 January 2019	Added Modbus Online in General Status Added Inhibit functionality
1.2	6 February 2019	Added parameter for inhibit when Communication is lost
1.3	8 January 2020	Added option to disable reading of Error register of Slave on connection establishment. Added option to force CANopen Router in Slave mode to startup in operational mode.

1. PREFACE

1.1. INTRODUCTION TO THE CANOPEN ROUTER

This manual describes the installation, operation, and diagnostics of the Aparian CANopen Router module. The CANopen Router, (hereafter referred to as the **module**,) provides intelligent data routing between either EtherNet/IP or Modbus TCP 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 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 Logix or Modbus devices in an existing CANopen network (by using the CANopen Router in Slave mode).

In a Logix system the module uses Direct-To-Tag technology allowing CANopen devices to exchange data with a Logix controller without the need to write any ladder or application code in Studio 5000.

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

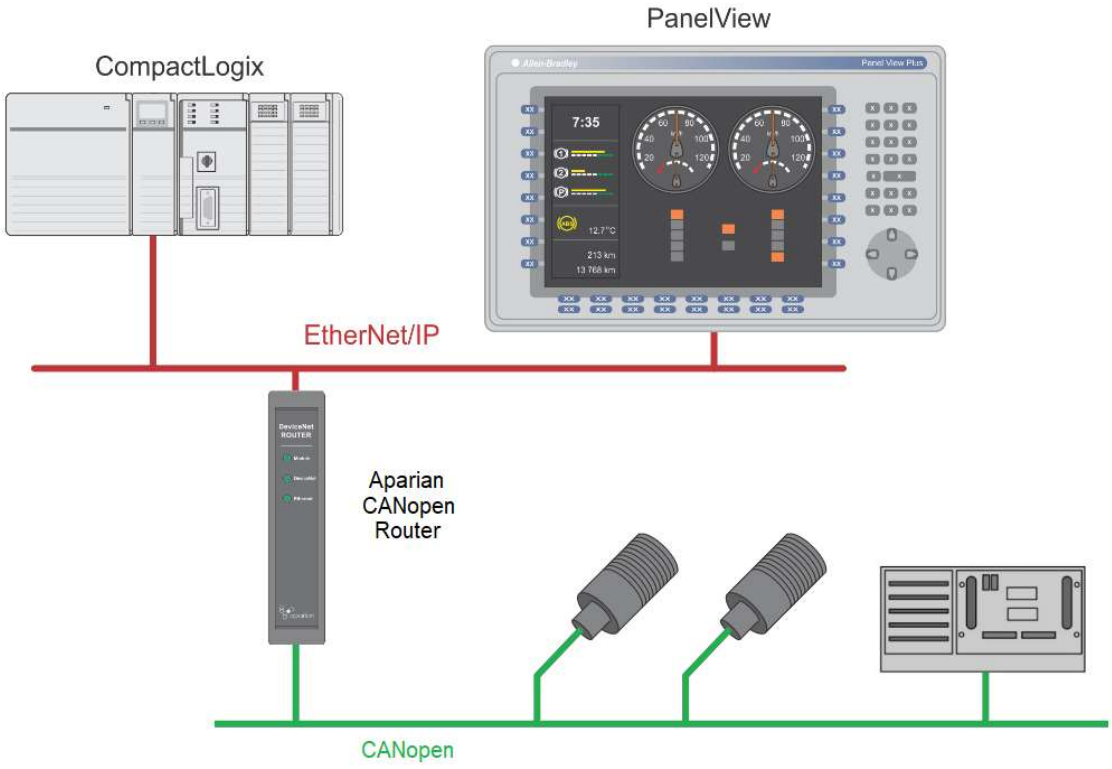


Figure 1.1. – Typical architecture using the CANopen Router

Slate allows the user to map up to 16 PDOs per CANopen Slave to Logix tags which will automatically be updated using Direct-To-Tag. When operating as a Modbus TCP Slave the module will provide various Modbus Holding registers to allow data exchange with a CANopen Slave.

Slate will also provide the user with the ability to change all parameters (using SDOs) of the slave based on the EDS file.

The module also provides a range of statistics to simplify the diagnostic process as well as a CANopen packet capture for remote diagnosis.

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.
- Module can interface to EtherNet/IP as well as Modbus TCP.
- Supports up to 64 CANopen Slaves (when in Master mode).
- Support for up to 16 PDOs (receive and transmit) per CANopen Slave.
- When using a Logix controller the module supports Direct-To-Tag so no Logix coding is required.
- Slate software provides a CANopen packet capture for better diagnosis of issues.
- Supports CiA 443 Bootloader Auto-enable.
- In Master Mode supports NMT message to initialize network.
- Time Synchronization of the CANopen network.
- Master supports SYNC for PDO communication.
- Supports all error and emergency (EMCY) messages and handling.
- Small form factor – DIN rail mounted.

1.3. ARCHITECTURE

The figure below provides an example of the typical network setup for connecting various CANopen Slaves to a Logix controller via the CANopen Router.

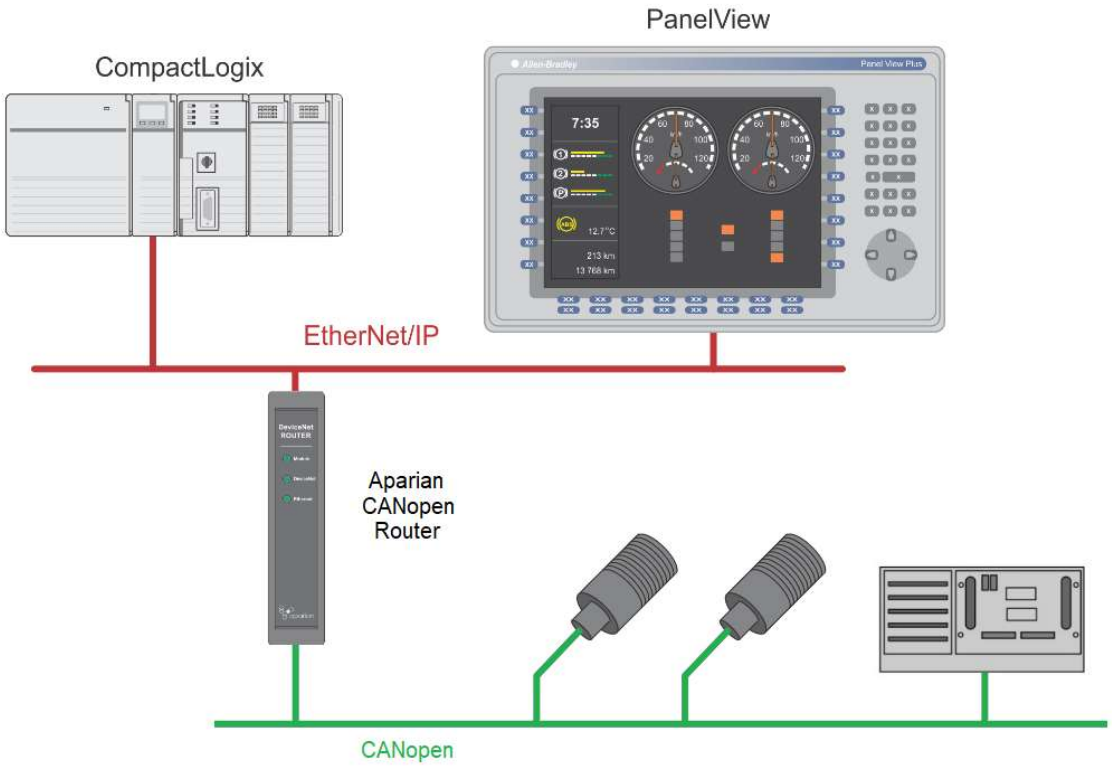


Figure 1.2. – Typical network setup for connecting CANopen Slaves to a Logix Controller

The same applies for interfacing CANopen Slaves to a Controller using Modbus TCP (as shown below).

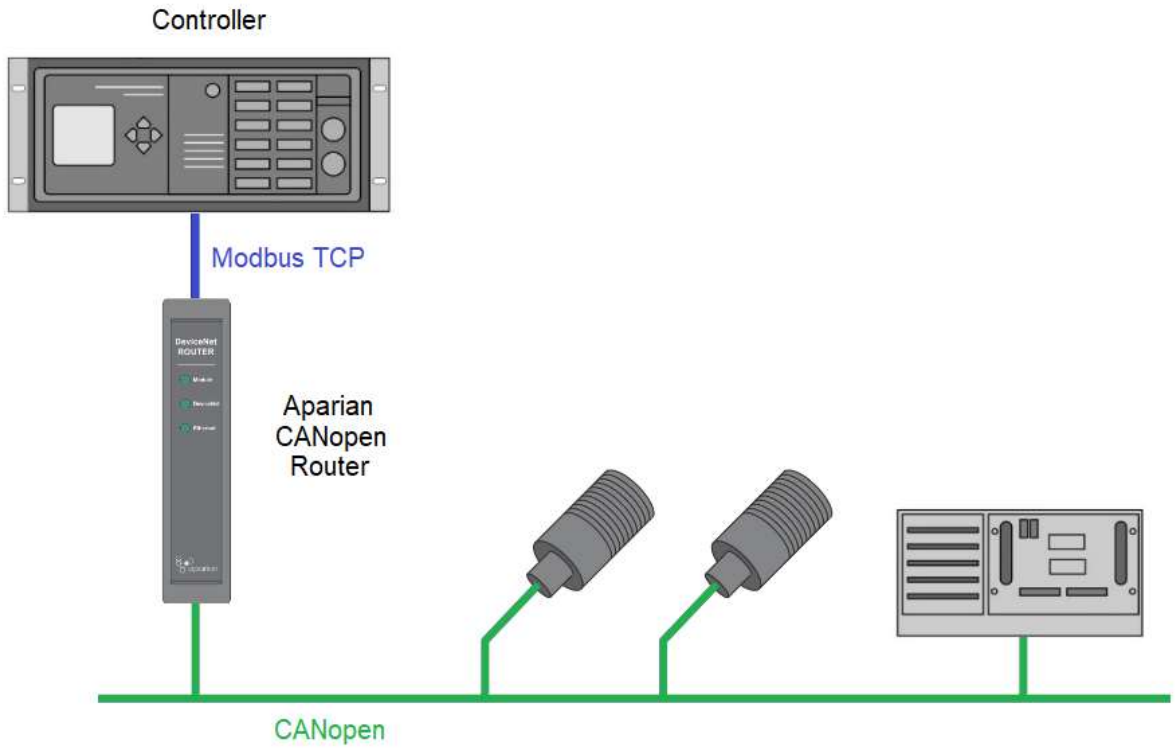


Figure 1.3. - Typical network setup for connecting CANopen Slaves to a Modbus Master

The next examples illustrate how the CANopen Router can be used as a CANopen Slave to allow Modbus devices and Logix controllers to integrate into an existing CANopen network.

Below is a typical network when the user is planning to use a Modbus device on an existing CANopen network using the CANopen Router.

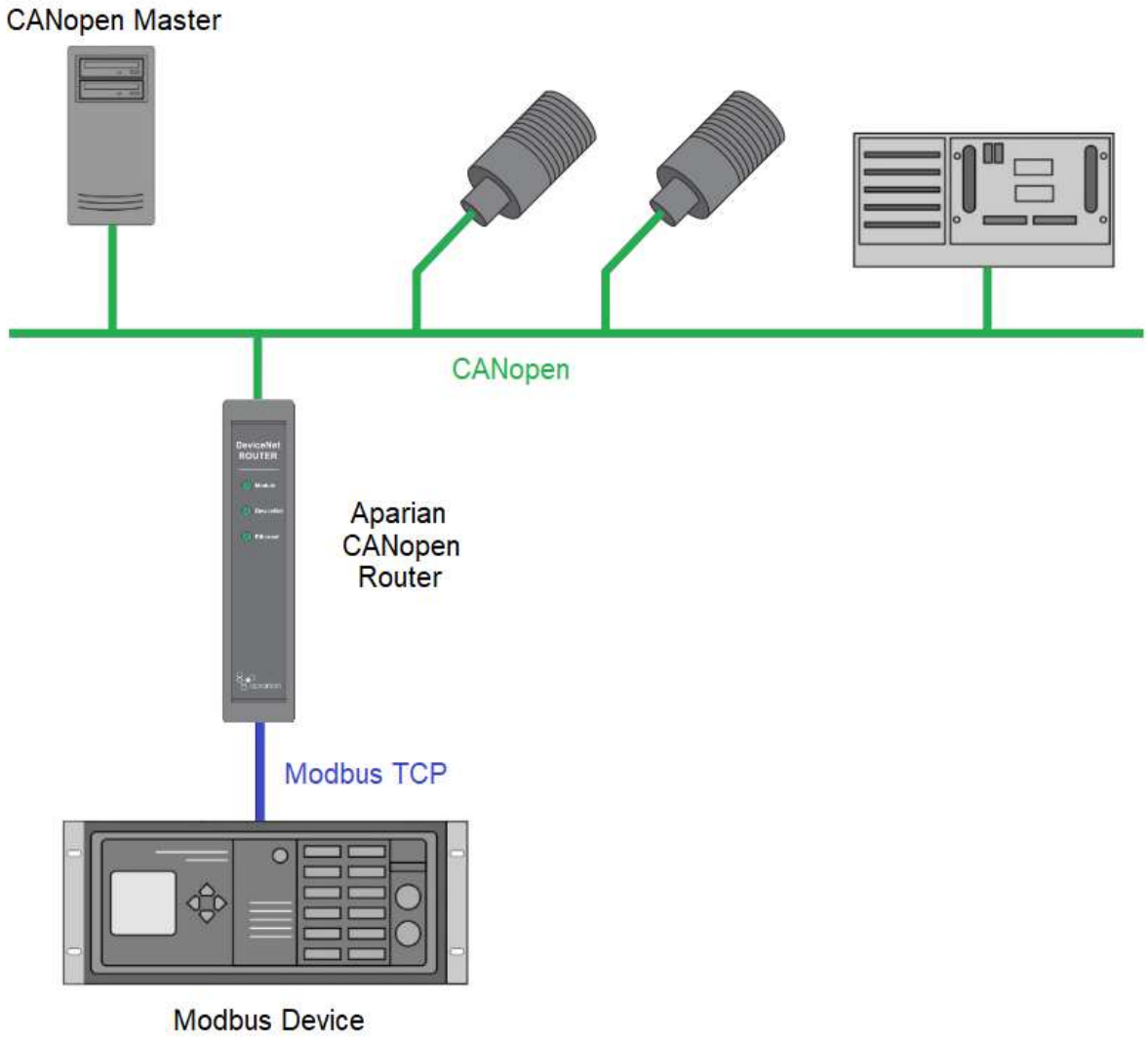


Figure 1.4. – Modbus Device acting as a CANopen Slave via the CANopen Router

Below is a typical network when the user is planning to use a Logix controller on an existing CANopen network using the CANopen Router.

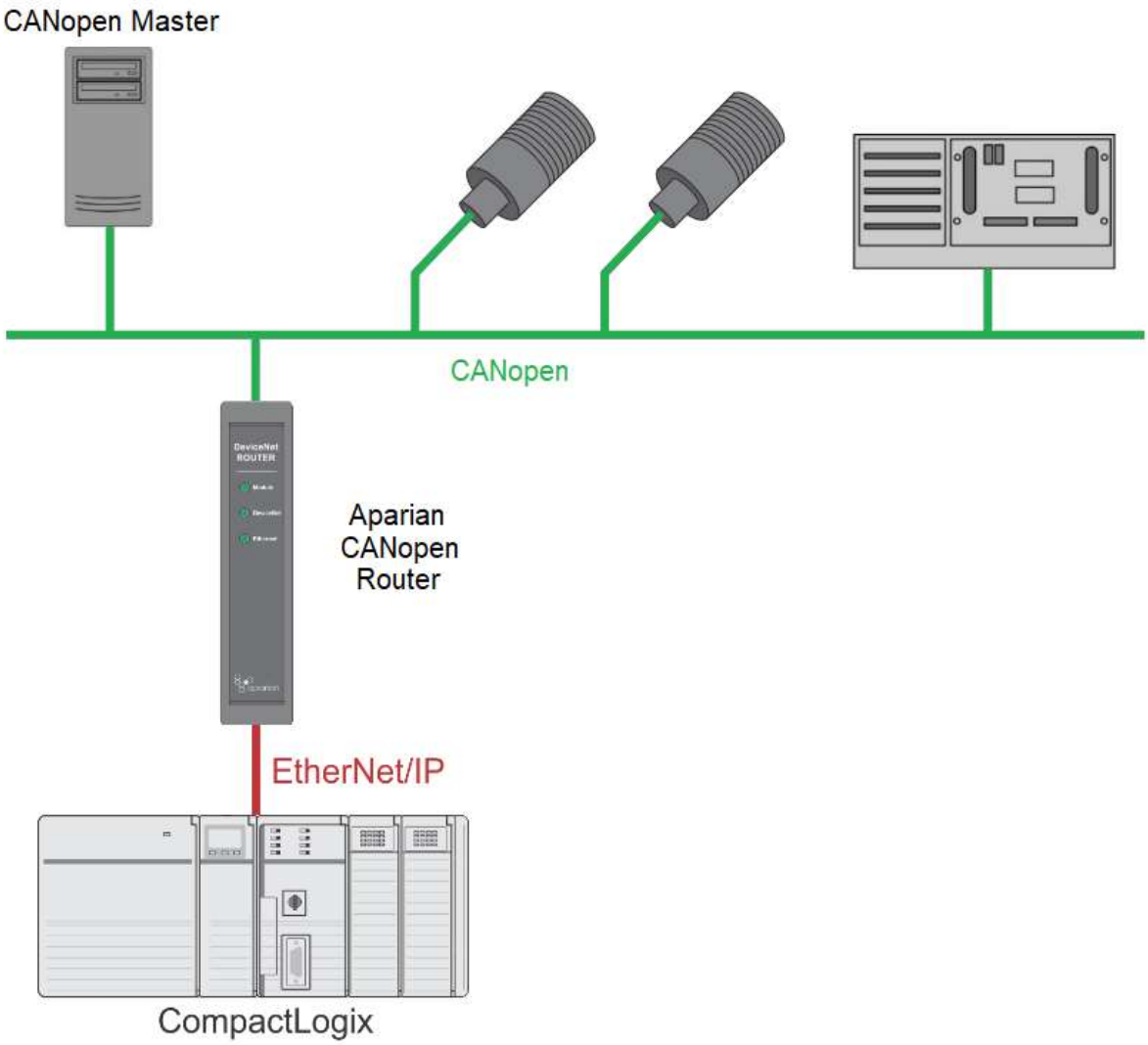


Figure 1.5. – Logix Controller acting as a CANopen Slave via the CANopen Router

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 User Manual CANopen Router Datasheet Example Code & UDTs	http://www.aparian.com/products/CANopenRouter
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	www.aparian.com/contact-us
Support email	support@aparian.com

Table 1.2. – Support Details

2. INSTALLATION

2.1. MODULE LAYOUT

The module has two ports at the bottom of the enclosure as shown in the figure below. The ports are used for Ethernet and CANopen. The 5-way connector also provides power to the module. The Ethernet cable must be wired according to industry standards which can be found in the additional information section of this document.

Figure 2.1. – CANopen Router side and bottom view

The module provides three diagnostic LEDs as shown in the front view figure below. These LEDs are used to provide information regarding the module system operation, the Ethernet interface, and the CANopen interface.

Figure 2.2. – CANopen Router front and 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 a 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	Reserved
DIP Switch 4	Applies the 120Ω terminating resistor across the CAN network (switched between Can-H and Can-L). **NOTE: When the module is at the start or the end of the CANopen network the terminator must be switched on.**

Table 2.1. - DIP Switch Settings

2.2. MODULE MOUNTING

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

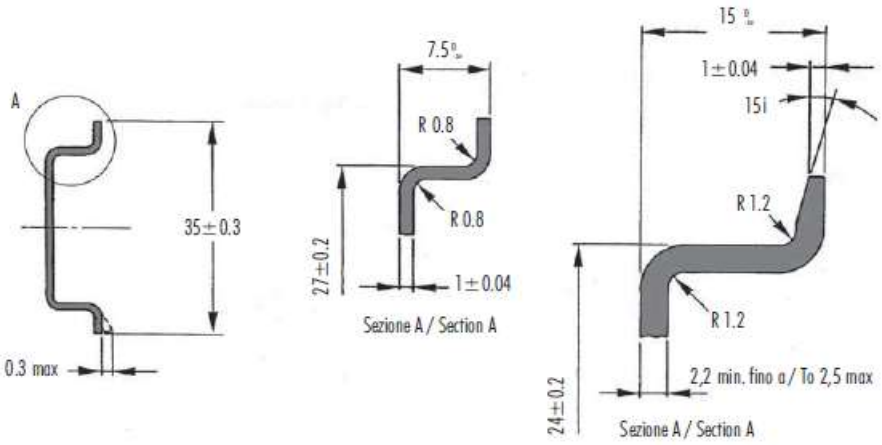


Figure 2.3 - DIN rail specification

The DIN rail clip is mounted on the bottom of the module at the back as shown in the figure below. Use a flat screw driver 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

2.3. CANOPEN AND POWER

A five-way CANopen 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 – 28Vdc. **Refer** to the technical specifications section in this document.

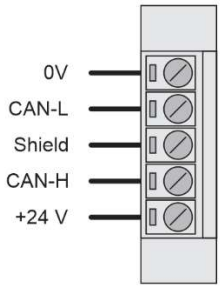


Figure 2.5 – CANopen and Power connector

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

2.4. ETHERNET PORT

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

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.

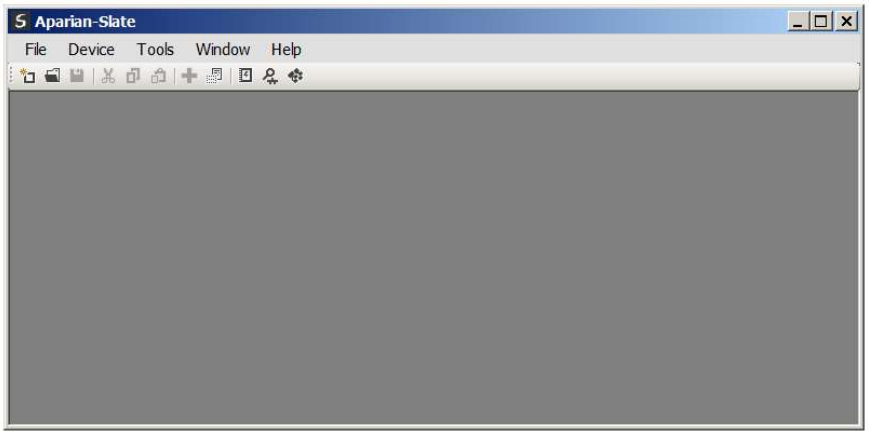


Figure 3.1. - Aparian Slate Environment

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.

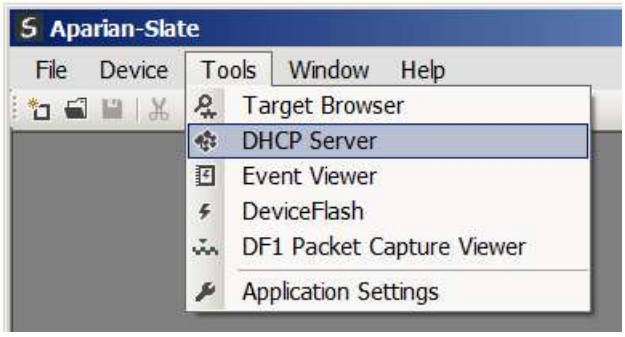


Figure 3.2. - Selecting DHCP Server

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

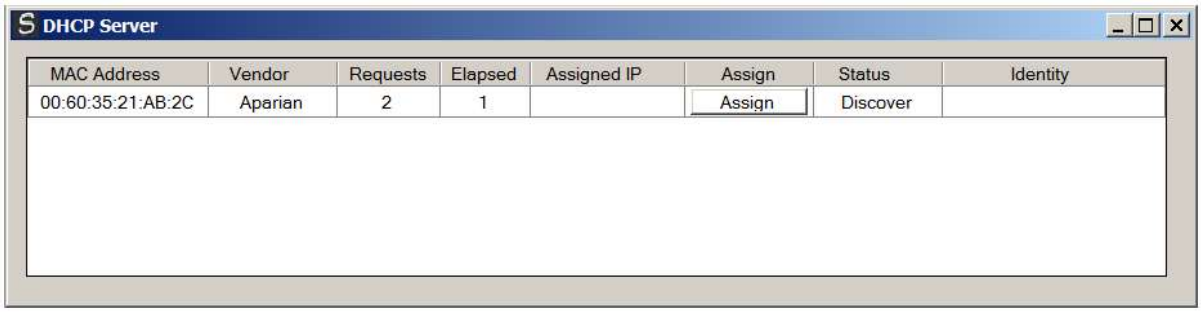


Figure 3.3. - DHCP Server

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.

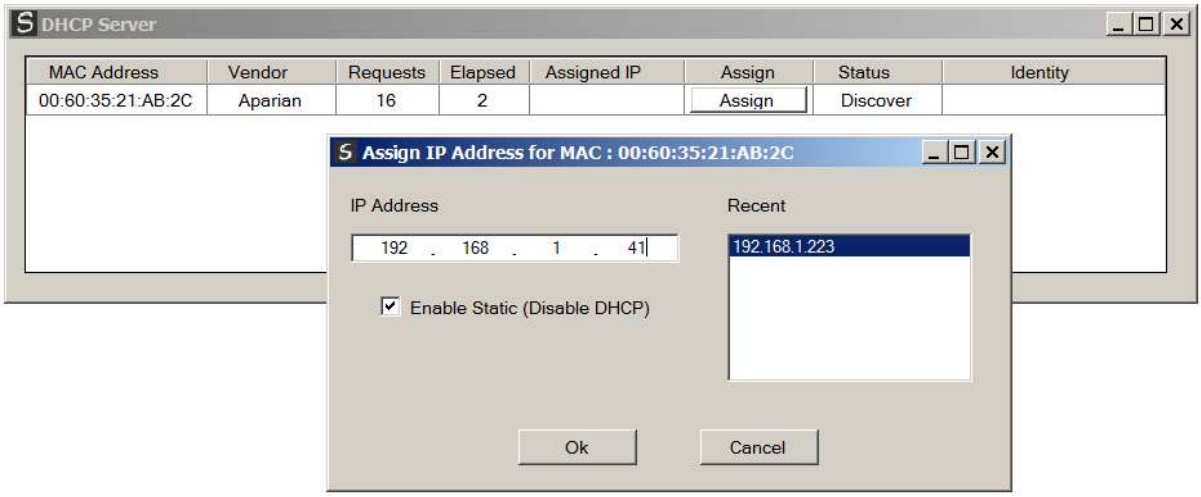


Figure 3.4. - Assigning IP Address

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.

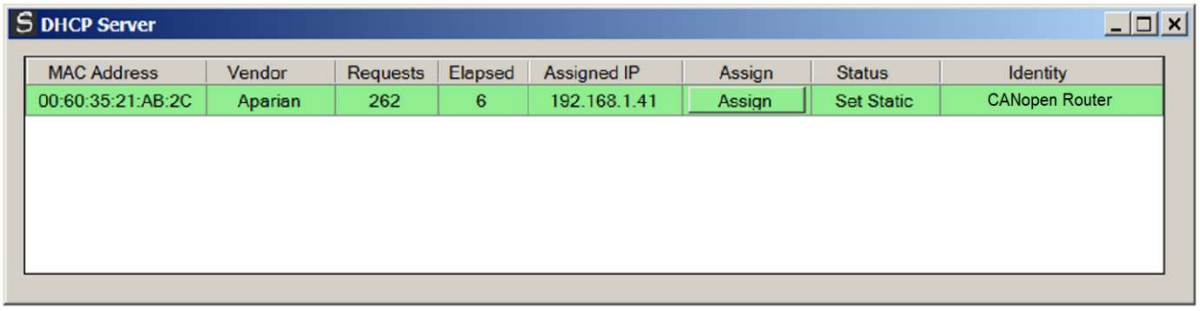


Figure 3.5. - Successful IP address assignment

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.

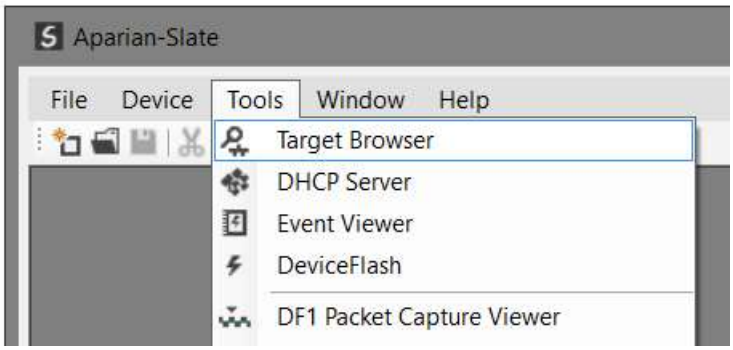


Figure 3.7. - Selecting the Target Browser

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

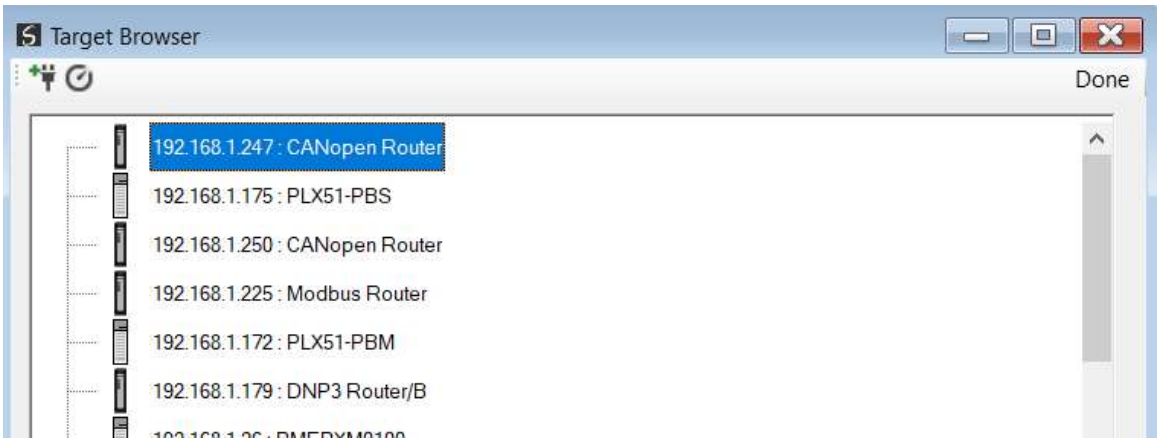


Figure 3.8. - Target Browser

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

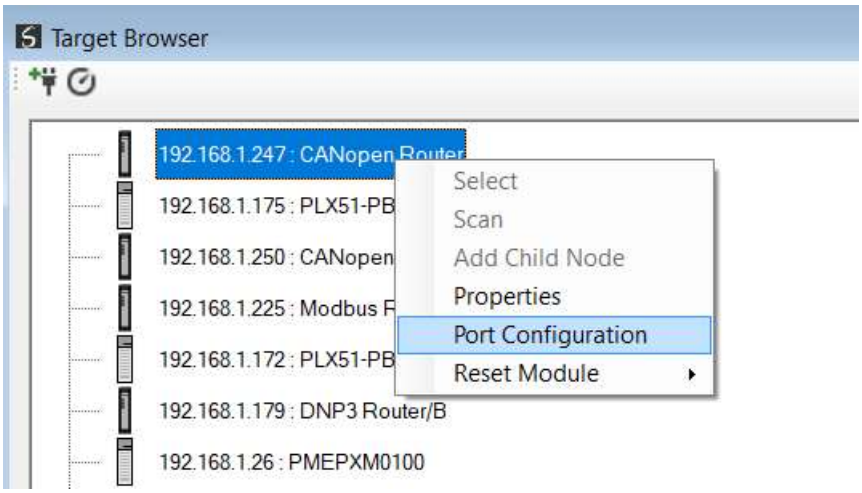


Figure 3.9. - Selecting Port Configuration

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

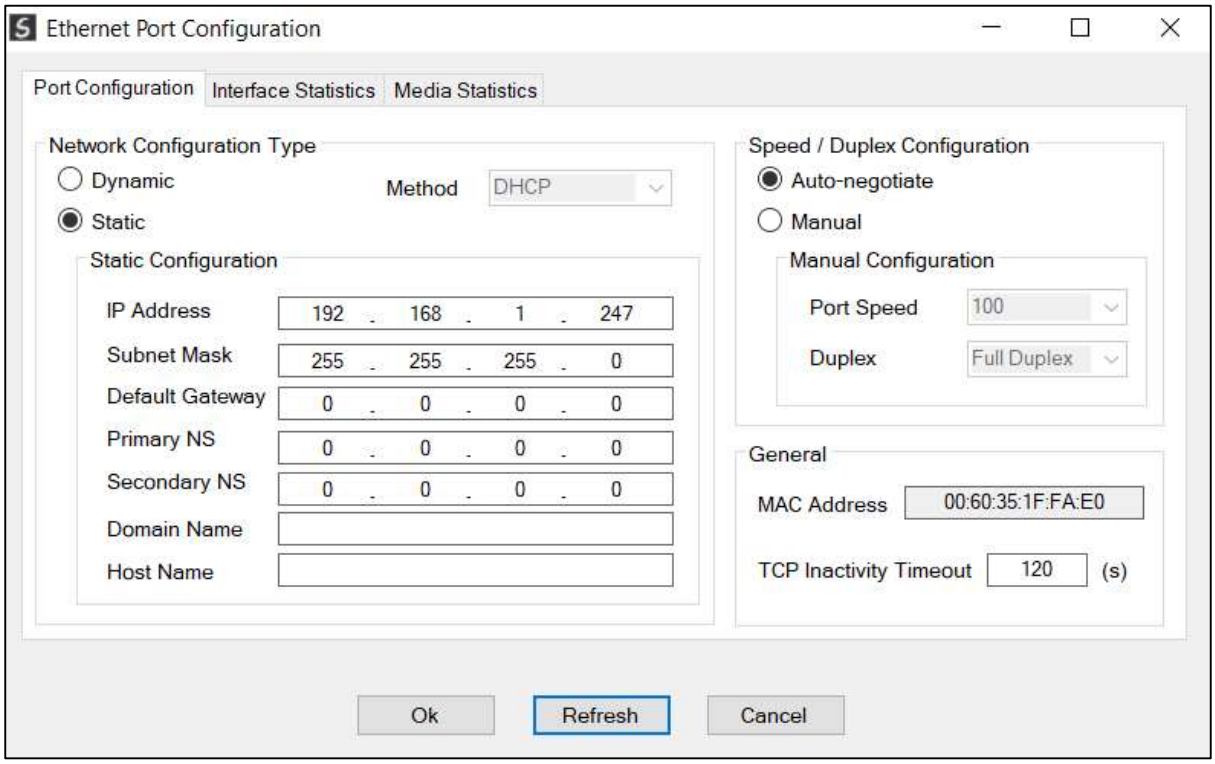


Figure 3.10. - Port Configuration

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.

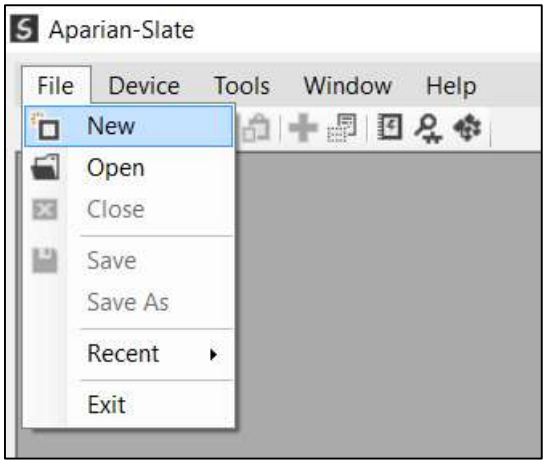


Figure 3.11. - Creating a new project

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.

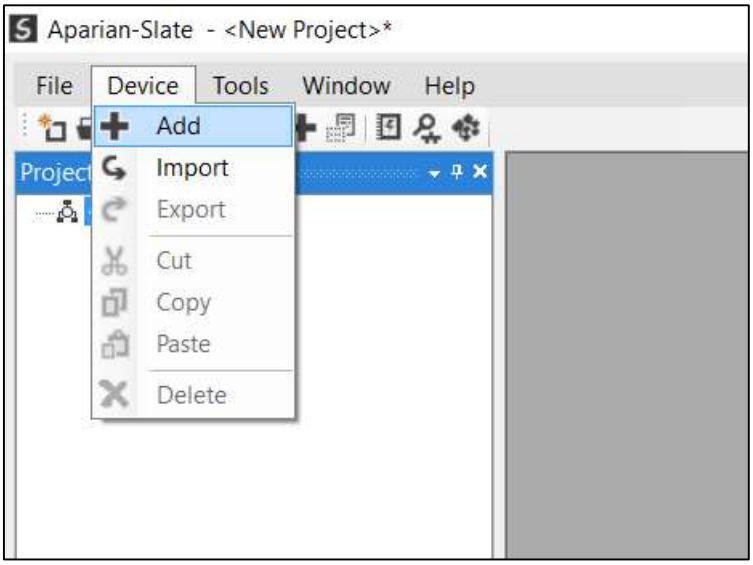


Figure 3.12. - Adding a new device

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

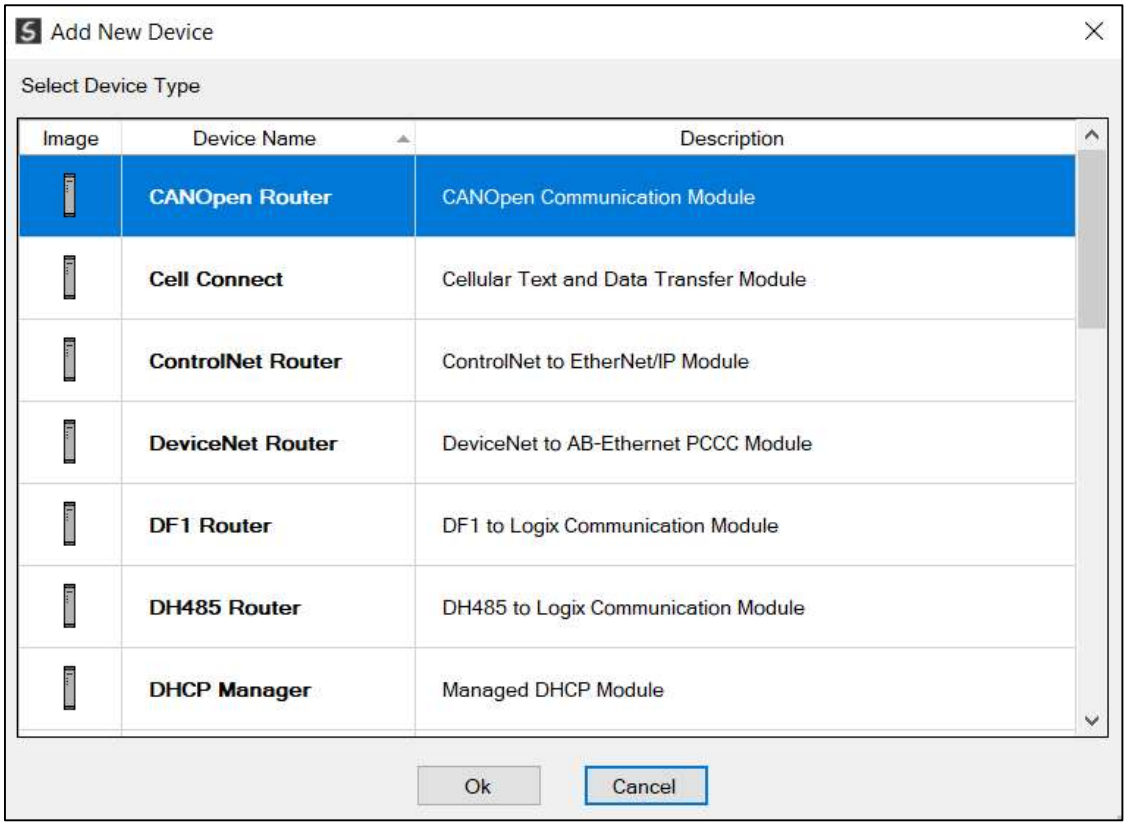


Figure 3.13 – Selecting a new CANopen Router

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*.

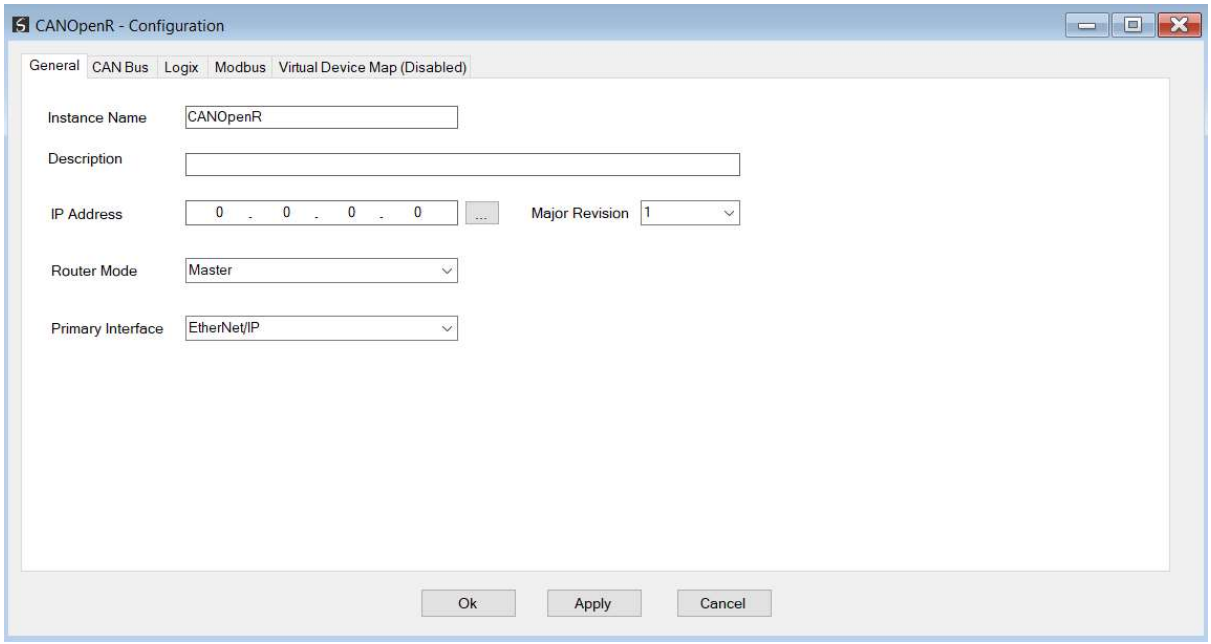


Figure 3.14. – CANopen Router configuration

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

3.4. CANOPEN ROUTER PARAMETERS

The CANopen Router parameters are configured using Slate. **Refer** to the additional information section for documentation and installation links for Aparian Slate. The CANopen Router parameter configuration consists of a general configuration, CAN Bus configuration, Logix, Modbus, 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 configuration and routing maps.

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

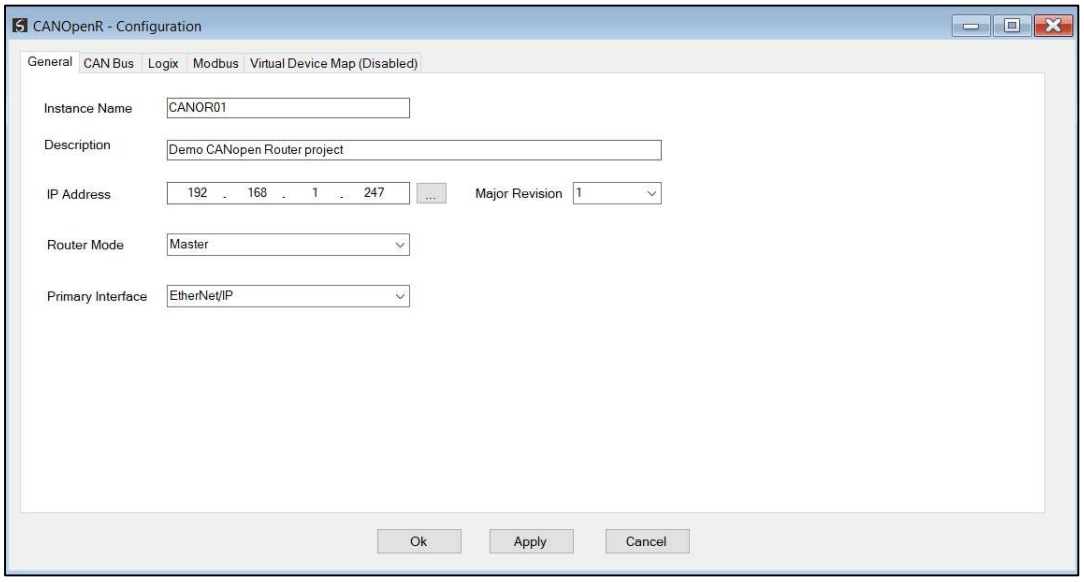


Figure 3.15. - General Configuration

The general configuration consists of the following parameters:

Parameter	Description
Instance Name	This parameter is a user defined name to identify between various CANopen Router modules.

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 on the network.
Major Revision	The major revision of the module
Router Mode	**Master** The CANopen Router will operate as a CANopen Master on the CANopen network. **Slave** The CANopen Router will operate as a CANopen Slave on the CANopen network.
Primary Interface	**EtherNet/IP** The CANopen Router will be an EtherNet/IP target and exchange data with a Logix controller using Class 1 and Class 3 communication. **Modbus TCP Slave** The CANopen Router will be a Modbus TCP Slave and can exchange data with a Modbus TCP Master.

Table 3.1 - General configuration parameters

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*.

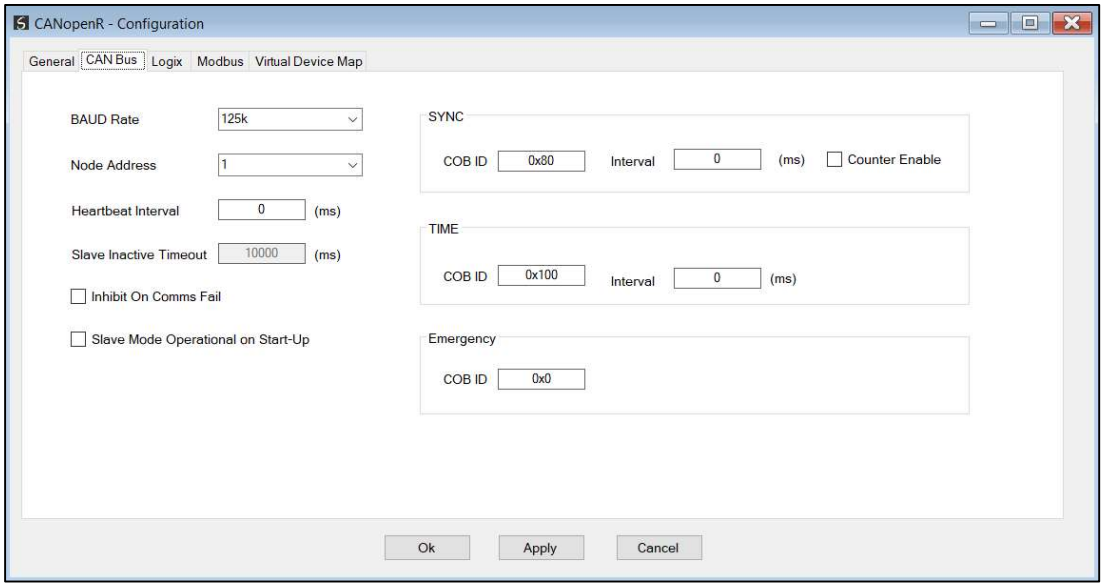


Figure 3.16 – CAN Bus Configuration

The CANopen Communication configuration consists of the following parameters:

Parameter	Description
BAUD Rate	The CANopen bus BAUD rate. The following options are available:

	• 50k • 125k • 250k • 500k • 800k • 1M
Node Address	The module's node address on the CANopen bus network.
Heartbeat Interval	This is the rate (in milliseconds) at which the CANopen Router will send out heartbeat messages on the CANopen bus. To disable sending of CANopen heartbeat messages the user can set this value to zero.
Slave Inactive Timeout	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. **(Master Mode Only)**
Inhibit On Comms Fail	The parameter will force the CANopen Router to inhibit the CANopen communication when either EtherNet/IP communication is lost, or Modbus TCP communication is lost.
Slave Mode Operational on Start-Up	When this is set the module (in slave mode) will start up in operational mode vs the normal pre-operational. **(Slave Mode Only)**
SYNC	**COB ID** The base address to use when sending and receiving SYNC messages on the CANopen network. **Interval** This is the rate (in milliseconds) at which the CANopen Router will send out SYNC messages on the CANopen bus. To disable sending of CANopen SYNC messages the user can set this value to zero. **Counter Enable** This will enable the optional parameter used to define and 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** This is the rate (in milliseconds) at which the CANopen Router will send out TIME messages on the CANopen bus. To disable 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

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

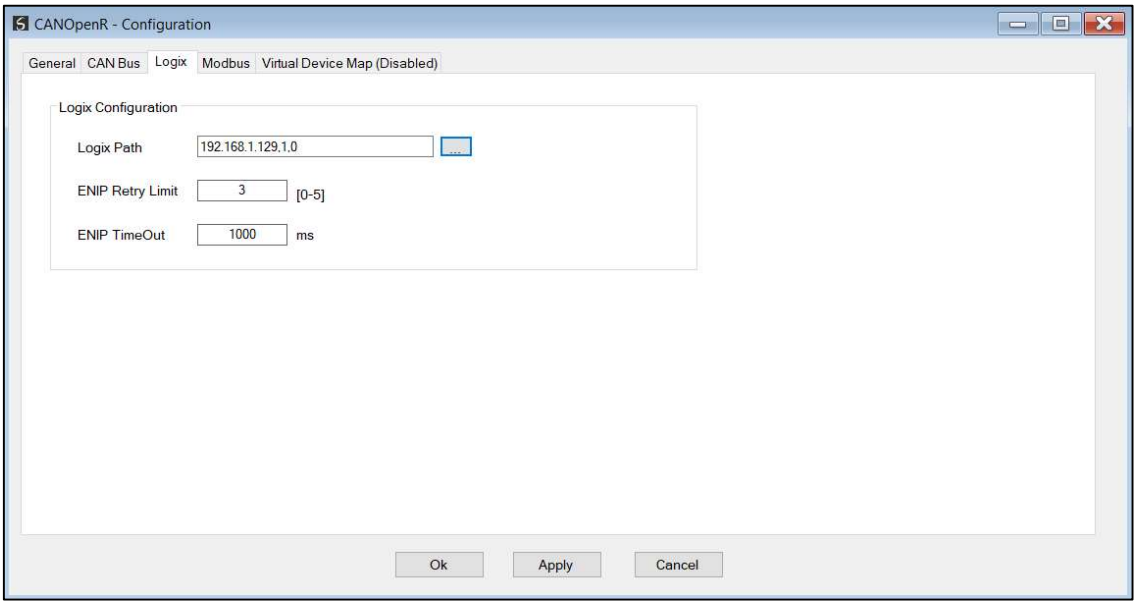


Figure 3.17 Logix Configuration

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 the select the Logix controller on the network.
ENIP Retry Limit	The amount of EtherNet/IP retries the module will make once no response was 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

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

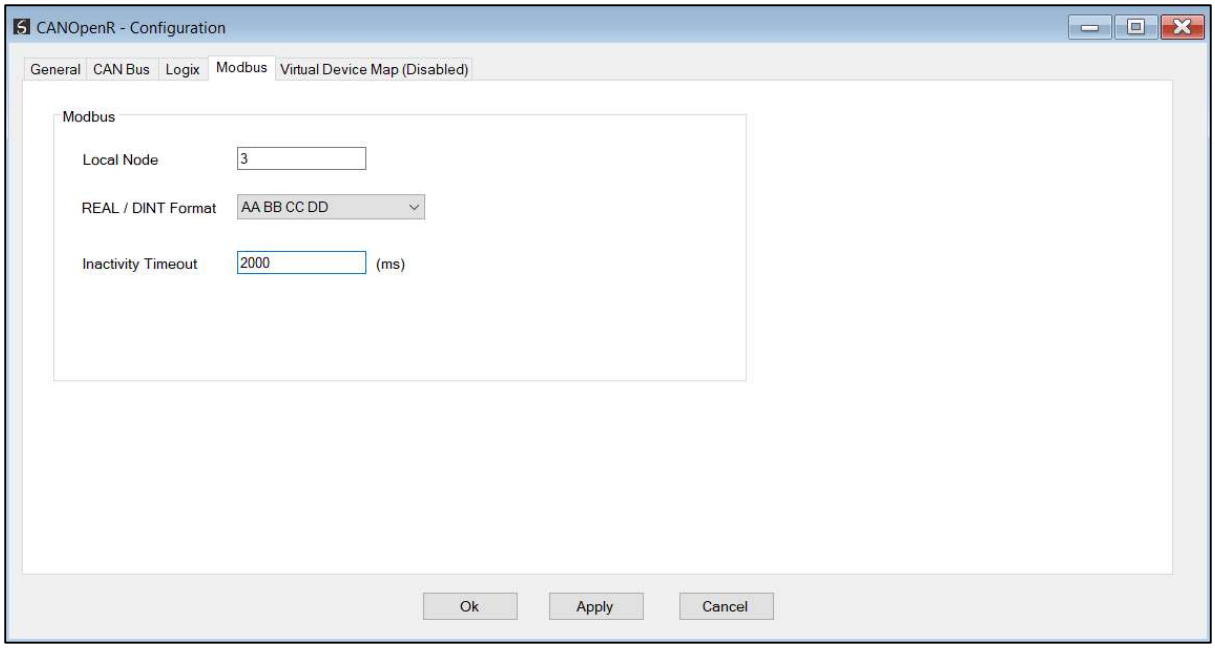


Figure 3.18 – Modbus Configuration

The Modbus Communication configuration consists of the following parameters:

Parameter	Description
Local Node	The Modbus Node address assigned to the CANopen Router.
REAL / DINT Format	For a Real (single floating point) number this setting shows the format of the data will be presented when using a Modbus Primary Interface. The format (byte re-ordering) options are as follows: • AA BB CC DD • BB AA DD CC • DD CC BB AA • CC DD AA BB
Inactivity Timeout	The amount of time that no Modbus requests have been received before the CANopen Router indicates that the connection to the Modbus Master is no longer intact.

Table 3.4 – Modbus parameters

3.5. CANOPEN MASTER MODE

The module can be configured to operate as a CANopen Master on the CANopen network (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 in Master mode.

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

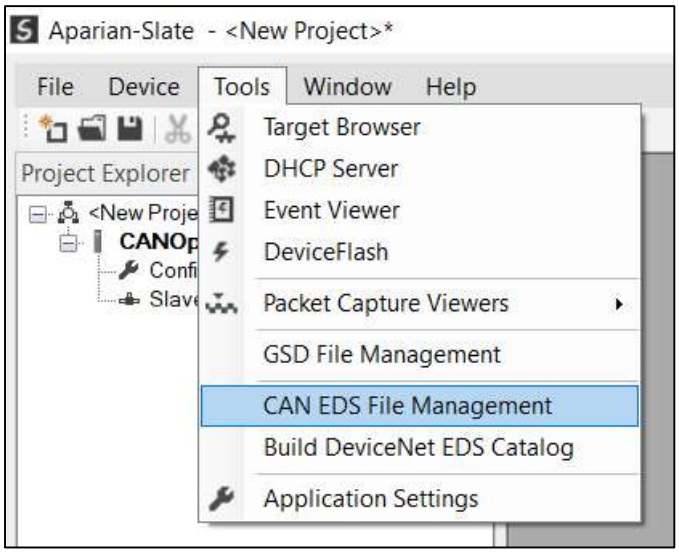
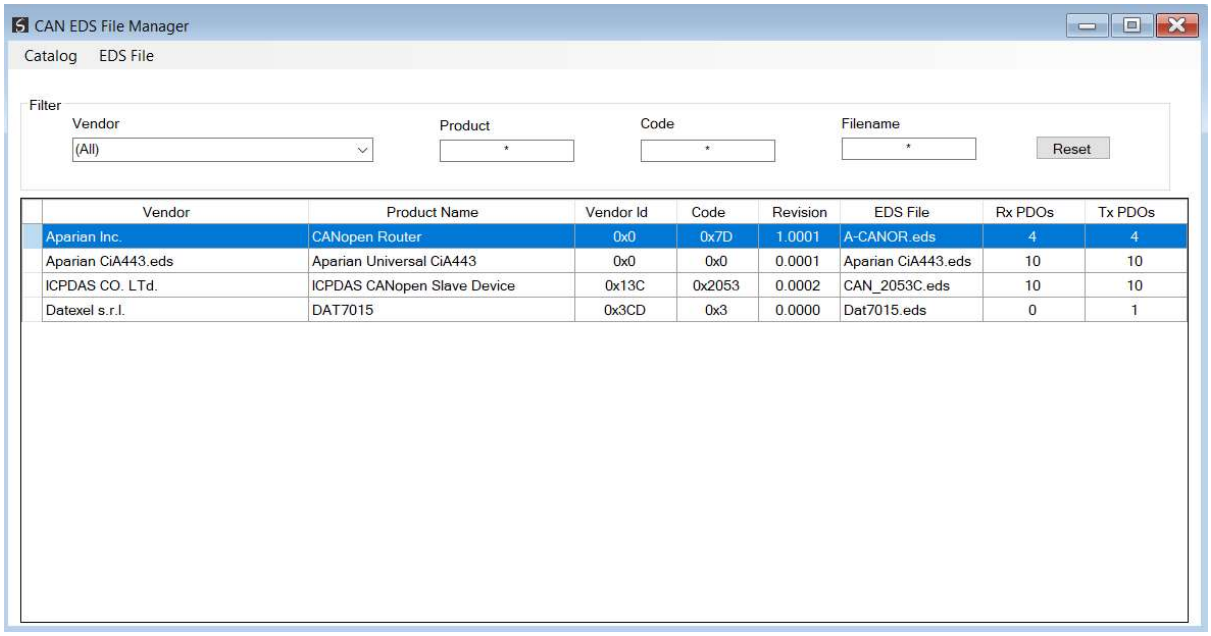


Figure 3.19 – Launching the CAN EDS File Management Tool

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



Vendor	Product Name	Vendor Id	Code	Revision	EDS File	Rx PDOs	Tx PDOs
Aparian Inc.	CANopen Router	0x0	0x7D	1.0001	A-CANOR.eds	4	4
Aparian CiA443.eds	Aparian Universal CiA443	0x0	0x0	0.0001	Aparian CiA443.eds	10	10
ICPDAS CO. LTd.	ICPDAS CANopen Slave Device	0x13C	0x2053	0.0002	CAN_2053C.eds	10	10
Datexel s.r.l.	DAT7015	0x3CD	0x3	0.0000	Dat7015.eds	0	1

Figure 3.20 – CAN EDS File Management Tool

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

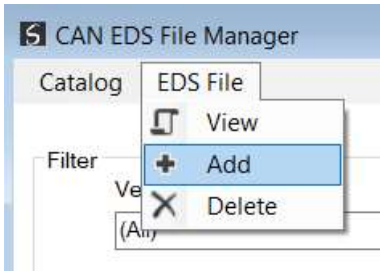


Figure 3.21 – CAN EDS File Adding

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

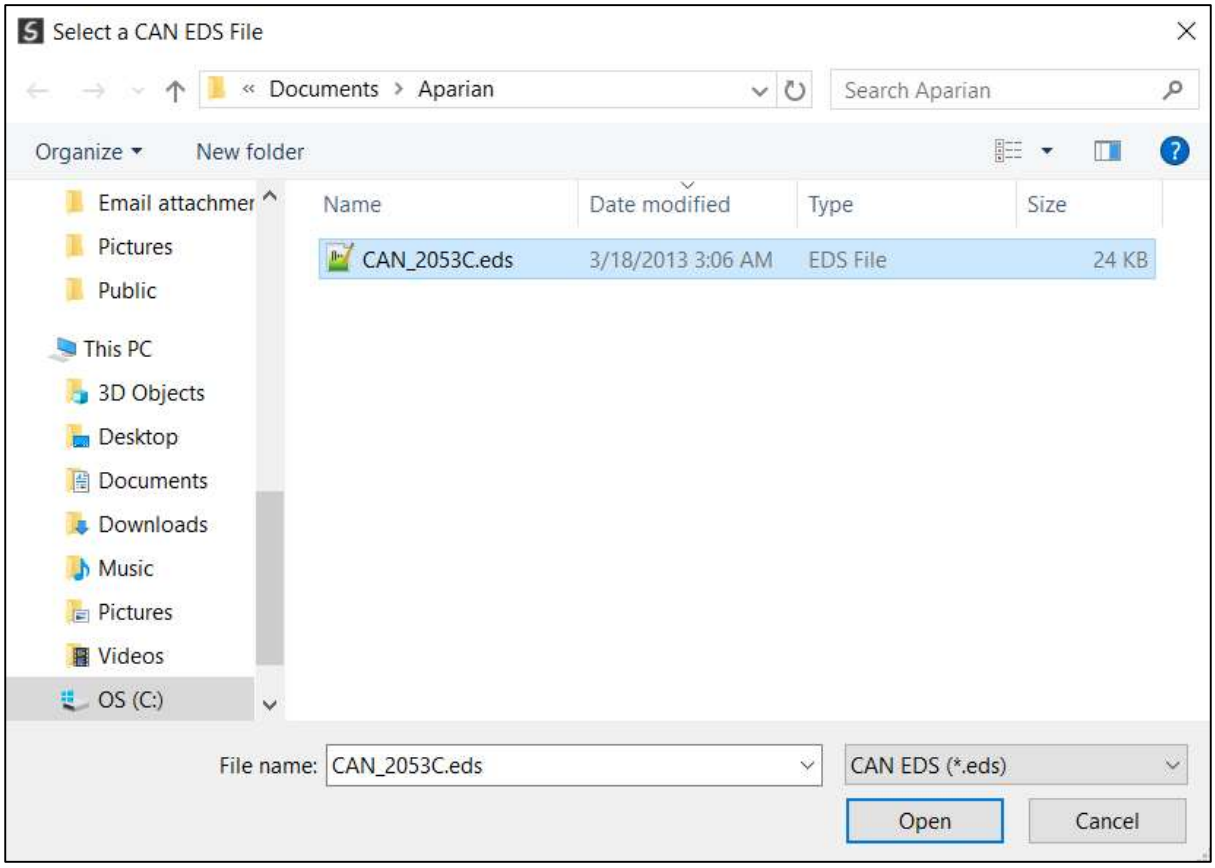


Figure 3.22 – CAN EDS File Adding

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.

A CAN EDS catalog can be exported from another Slate by exporting the CAN EDS catalog on one Slate and importing it in another. This is done by selecting either *Import* or *Export* under the Catalog menu as shown below:

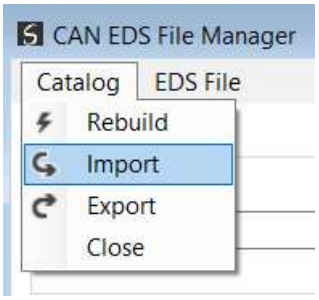


Figure 3.23 – CAN EDS Catalog importing

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 is done by right-clicking on the ***Slave Devices*** item in the tree and selecting ***Add CANopen Device***.

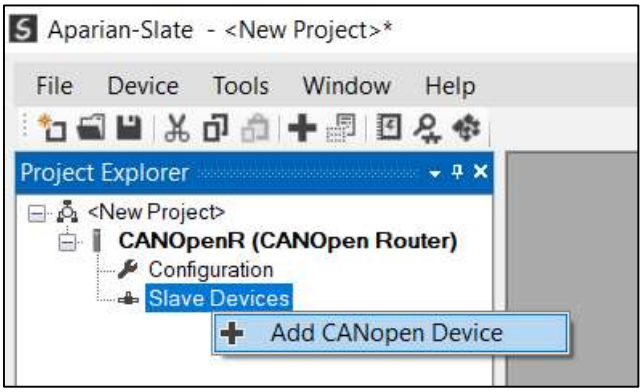


Figure 3.24 – Adding a CANopen Slave Device

3.5.3. General Configuration

The General configuration 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*.

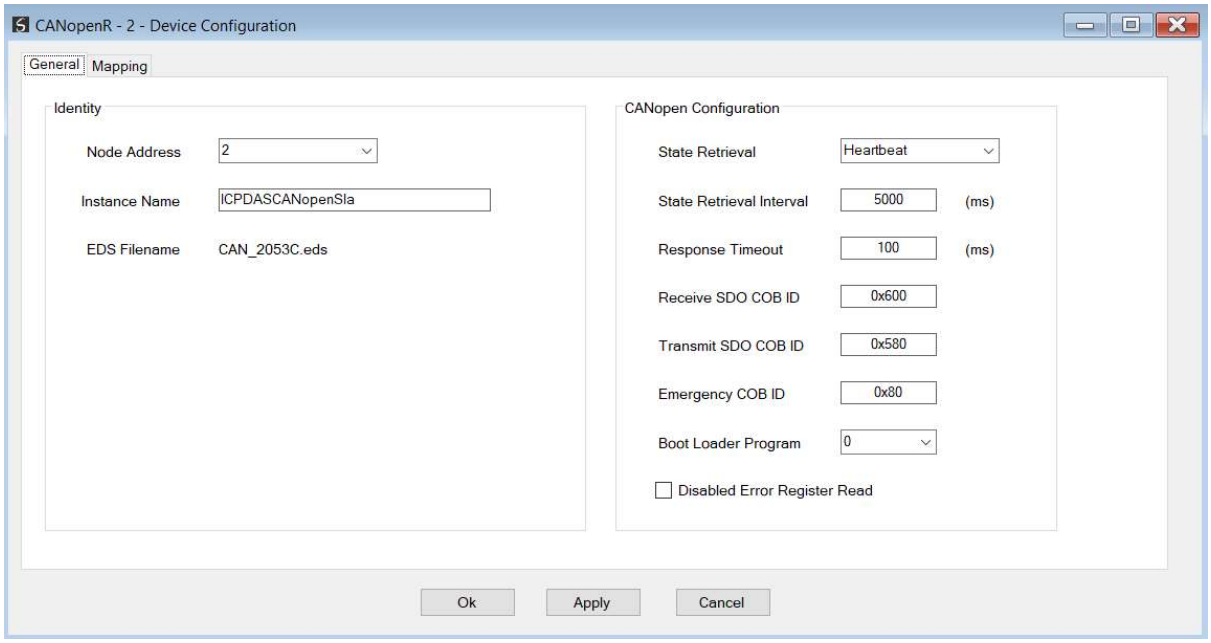


Figure 3.25 – Device General configuration parameters

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 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. 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 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 response to 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:** Slate will automatically add the CANopen slave device node address to this base value (so the user must enter the base address e.g. 0x600). As an example, for node address 2 the configured Receive SDO SOB ID will be 0x602 by default.
Transmit SDO COB ID	This is the base COB ID used for transmitting SDO messages from the CANopen Slave device. **NOTE:** Slate will automatically add the CANopen slave device node address to this base value (so the user must enter the base address e.g. 0x580). As an example, for node address 2 the configured Transmit SDO SOB ID will be 0x602 by default.
Emergency COB ID	This is the base COB ID used for transmitting Emergency (EMCY) messages from the CANopen Slave device. **NOTE:** Slate will automatically add the CANopen slave device node address to this base value (so the user must enter the base address e.g. 0x80). As an example, for node address 2 the configured Emergency SOB ID will be 0x82 by default.
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 enabled the CANopen Router in Master mode will not attempt to read the Error Register (0x1001) of the Slave device on connection establishment.

Table 3.5 –Device General configuration parameters

3.5.4. MAPPING

The module can be configured to exchange data between either a Logix controller or Modbus Master and various CANopen Slave devices.

When the primary interface is EtherNet/IP, the CANopen Router will allow the user to read data from a CANopen Slave device into a Logix controller and/or write data to a CANopen Slave device from a Logix controller.

When the primary interface is Modbus TCP Slave, the CANopen Router will allow the user to read data from a CANopen Slave device into a Modbus Holding Register and/or write data to a CANopen Slave device from a Modbus Holding Register.

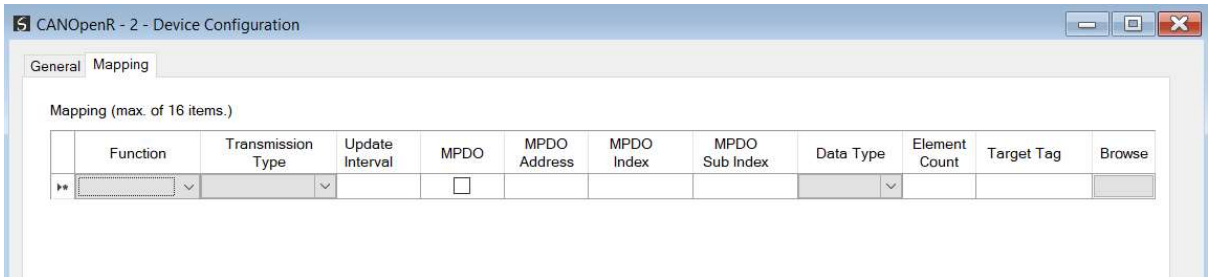


Figure 3.26 – Device Mapping parameters

Parameter	Description
Function	There are two functions supported for mapping field device PDOs (process variables). **TPDO x** 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 PDO received from the Slave device can be up to 8 bytes (e.g. two 32-bit Reals). **RPDO x** 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 PDO sent to the Slave device can be up to 8 bytes (e.g. two 32-bit Reals).
Transmission Type	**Sync (RPDO only)** The CANopen Router in Master mode will send out the PDO data to the CANopen slave once the SYNC packet has been sent. **RemoteTxReq (TPDO only)** The CANopen Router will request the PDO from the CANopen slave at the update interval. If the slave device has been configured to automatically send out the PDOs, it is recommended to set the update interval at 2 x the rate the PDOs will be sent. **Evt-Timer (RPDO only)** The CANopen Router in Master mode will send out the PDO data to the CANopen slave every Update Interval. **Evt-Logix (RPDO only)**

	The CANopen Router in Master mode will send out the PDO data to the CANopen slave every time the relevant PDO bit in the ***SlaveOutputTriggers*** of the Logix output assembly or Modbus Holding Register.
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 Logix Tag or Modbus Holding Register.
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 element count multiplied by the data type size must not be greater than 8 bytes when **not** using multiplexing and 4 bytes when using multiplexing.
Target Tag	When the Primary Interface is EtherNet/IP, this parameter will be the Logix Tag that will be used to exchange data with the specific CANopen slave device. The target tag can either be entered manually or if online with the controller the target tag can be updated using the target browser (see figure below).

Table 3.6 –Device Mapping parameters

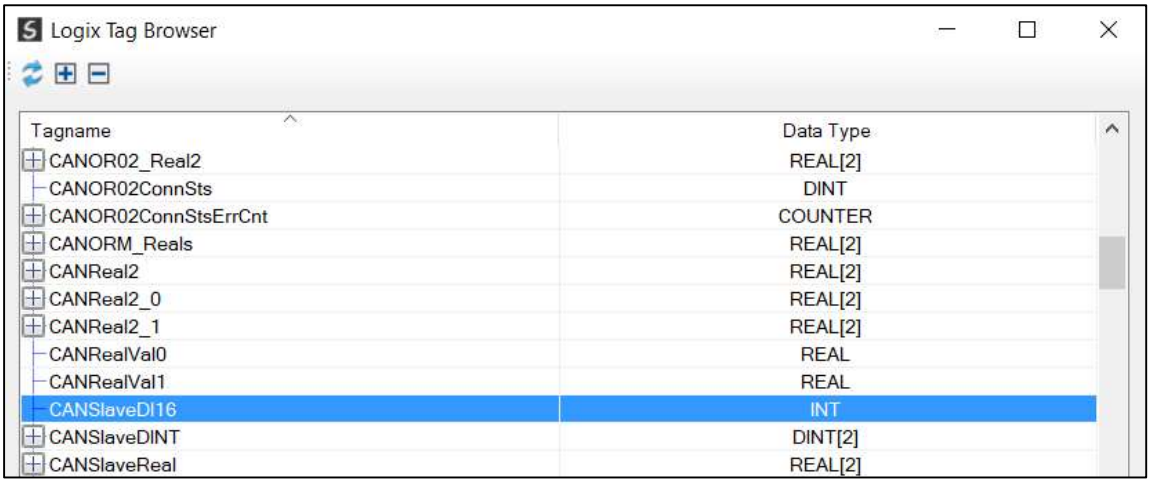


Figure 3.27 – Target Tag selection

3.5.4.1. ETHERNET/IP INTERFACE

When using the EtherNet/IP interface, 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.

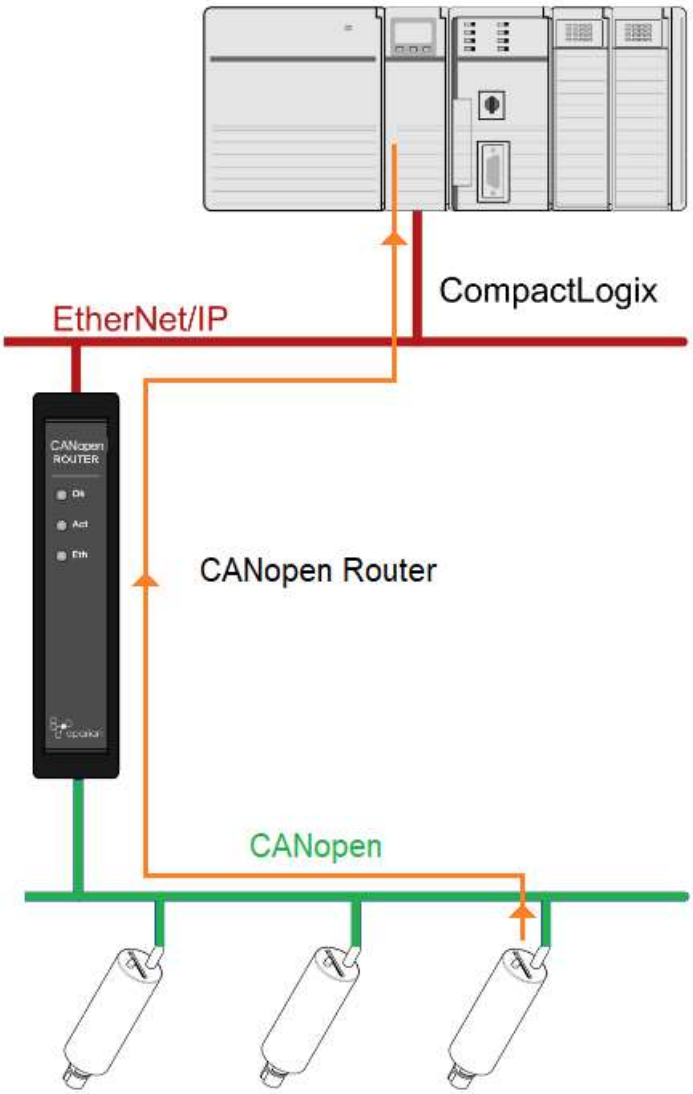


Figure 3.28 – Process variable (TPDO) from slave device to Target Tag

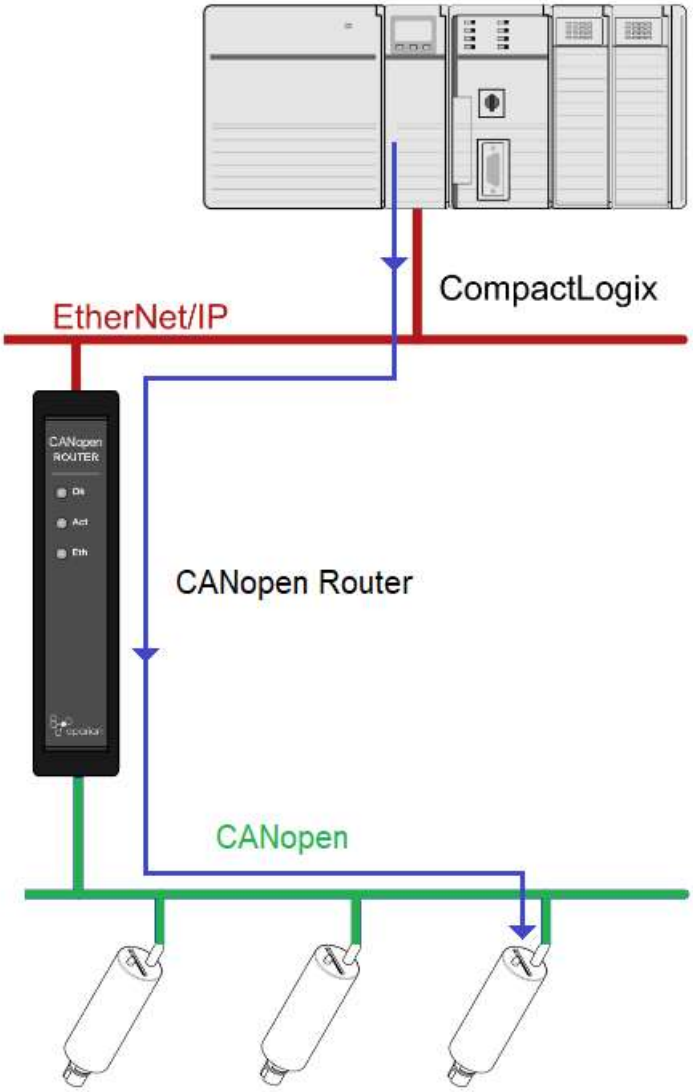


Figure 3.29 – Process variable (RPDO) from Target Tag to slave device

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 cannot be only one REAL.

NOTE: If there are duplicate mapping items in the mapping list then only the first mapped item (of all the duplicates) will be executed.

3.5.4.2. MODBUS TCP INTERFACE

When Modbus TCP has been selected as the primary interface, the process variables (TPDOs) from the CANopen Slave device will be stored in predefined Modbus Holding Register. The process variable (RPDOs) that will be sent to the CANopen Slave device will also be read from the predefined internal Modbus Registers.

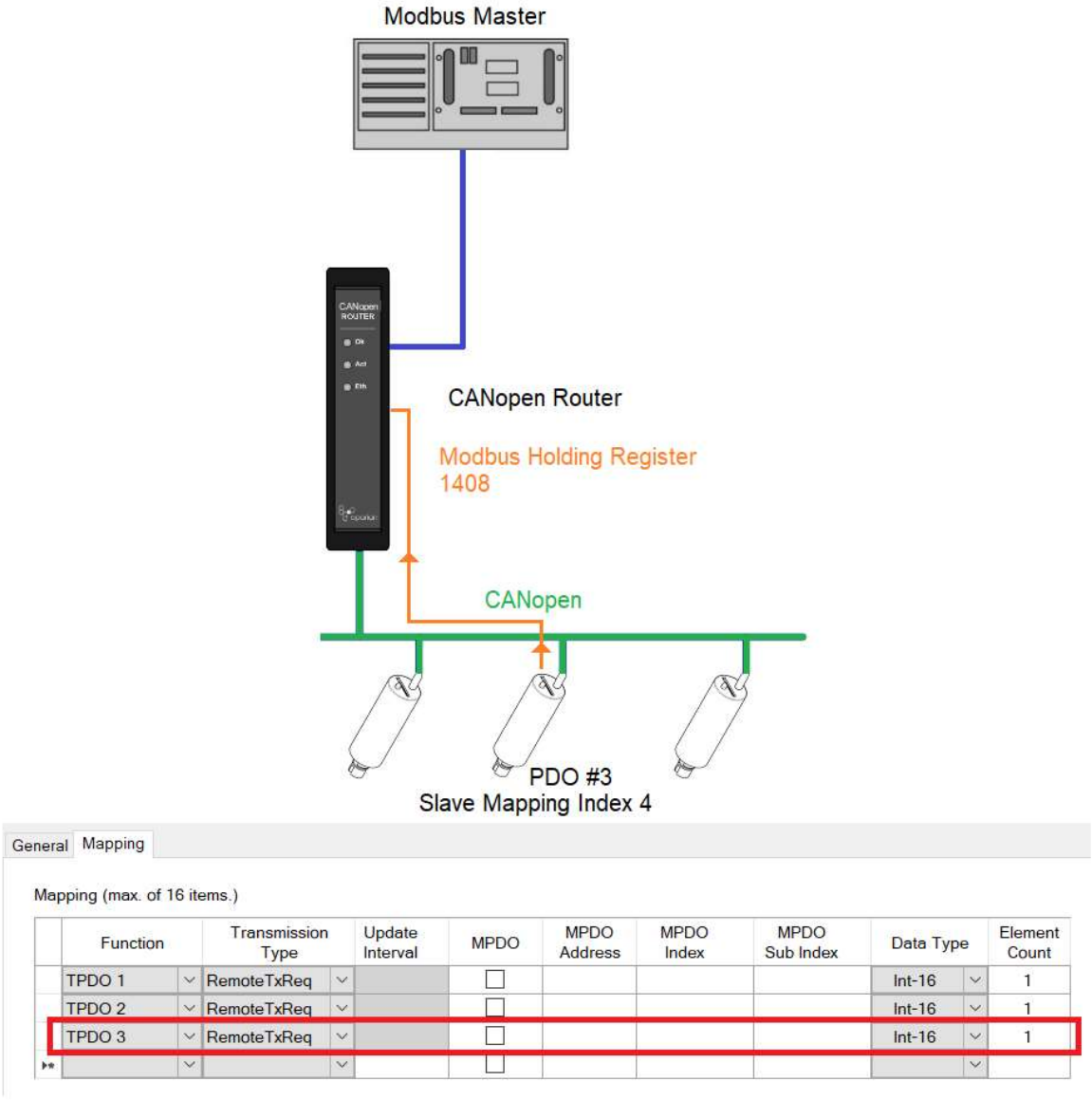


Figure 3.30 – Process variable (TPDO) from slave device to Modbus Holding Register

In the above example the PDO value will be written to Modbus Holding Register 1408. This is calculated as follow:

Modbus Holding Register
Slave Device start– *1000*
Slave Device 5 – *1000 + (Slave Mapping Index * 100) = 1400*
Slave Device 5 PDO 3 – *1400 + ((PDO -1) * 4) = 1408*

NOTE: There will be a 100-register gap between consecutive field devices. See the Modbus Mapping section for more details.

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

NOTE: The PDO offset in the Modbus Holding Register will depend on its location in the Mapping. For example, if a Slave Node with TPDO 3 at slave mapping index 4 (as shown below) will be at Modbus Holding Register 1416 *(1400 + (5 – 1) * 4)*. The offset in the Modbus Holding registers is independent of the TPDO number (e.g. TPDO **2**).

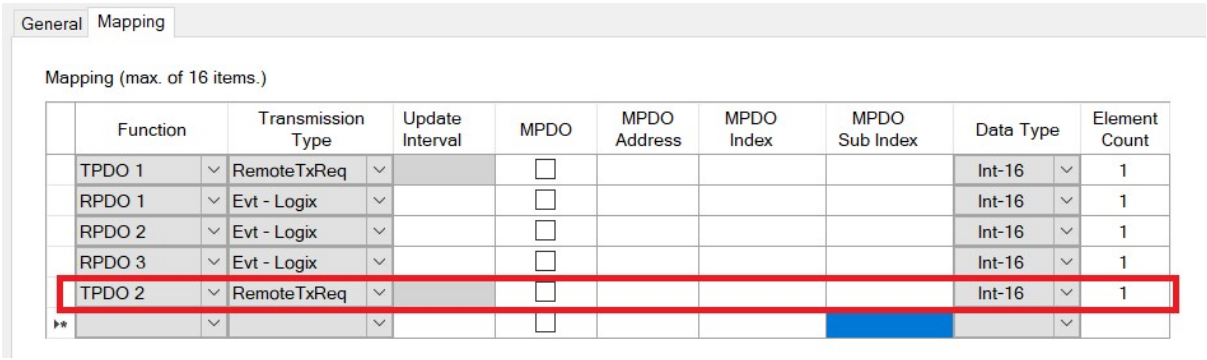


Figure 3.31 – Process variable (TPDO) at mapping index 5

NOTE: When sending process variables to the field device, the same example and calculation as above applies.

NOTE: To optimise the Modbus communication it is recommended to group all the TPDOs together and then all the RPDOs.

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.

3.5.5. 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):

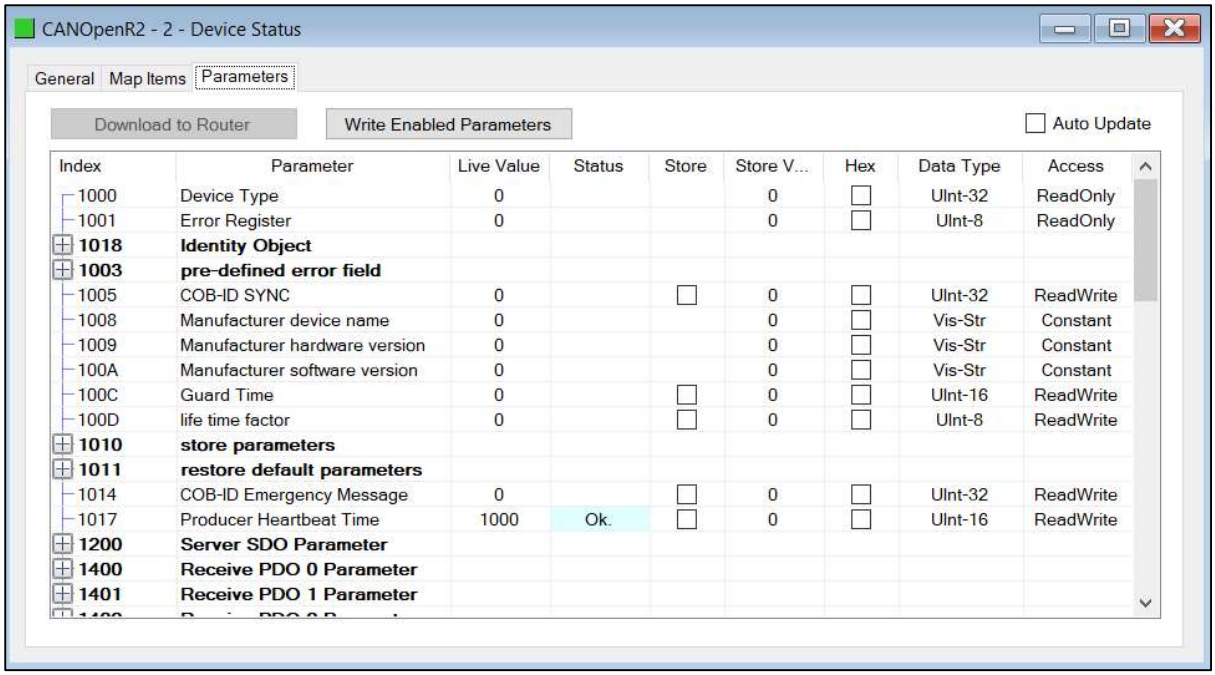


Figure 3.32 – Slave device parameters

These parameters will be listed from the EDS file used to instantiate the CANopen Slave device. 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.

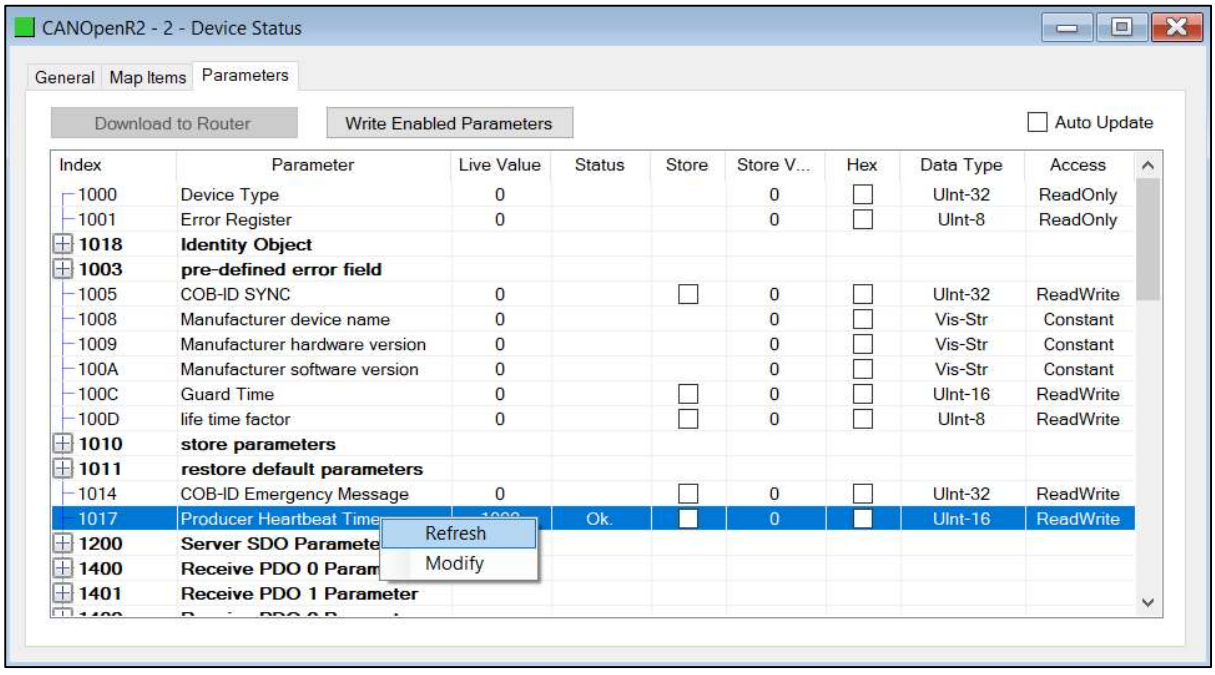


Figure 3.33 – Slave device parameters - Read

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

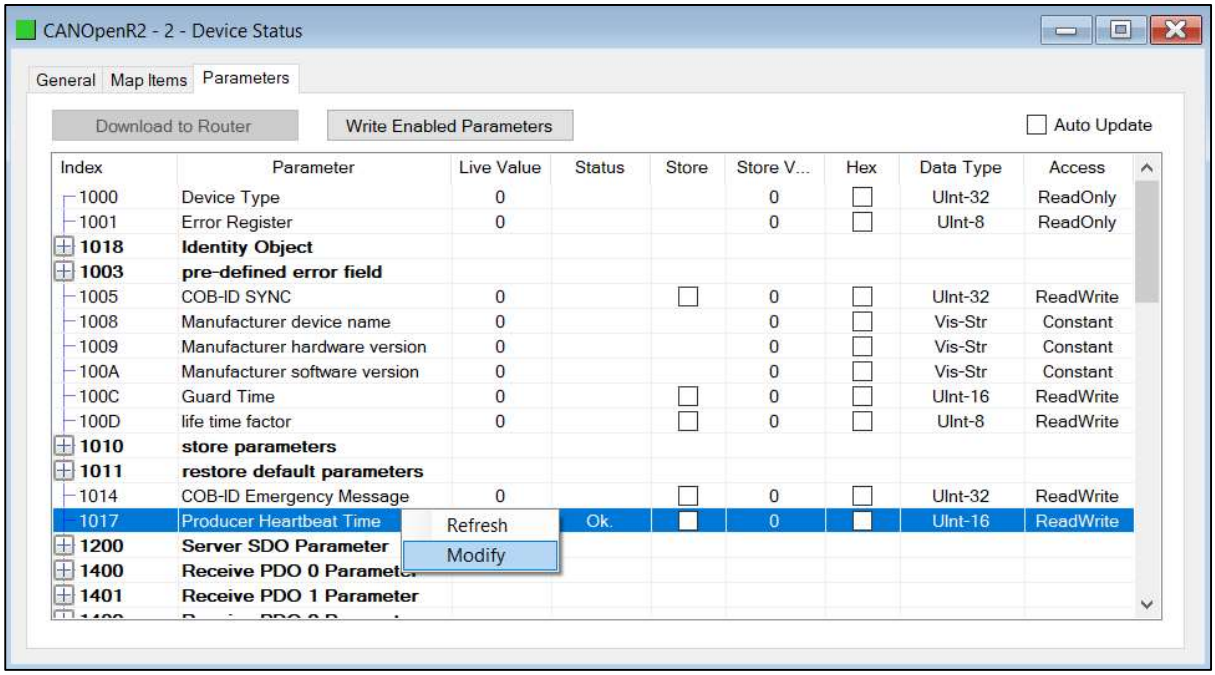


Figure 3.34 – Slave device parameters – Modify

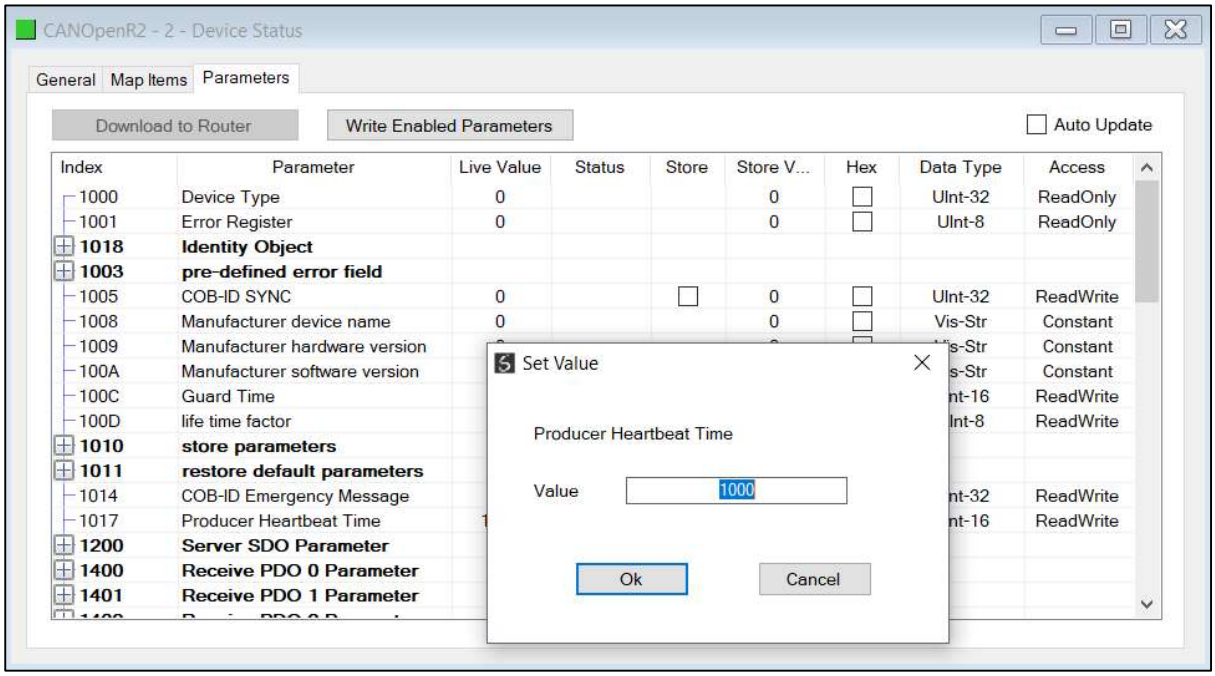


Figure 3.35 – Slave device parameters – Set Value

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* section).

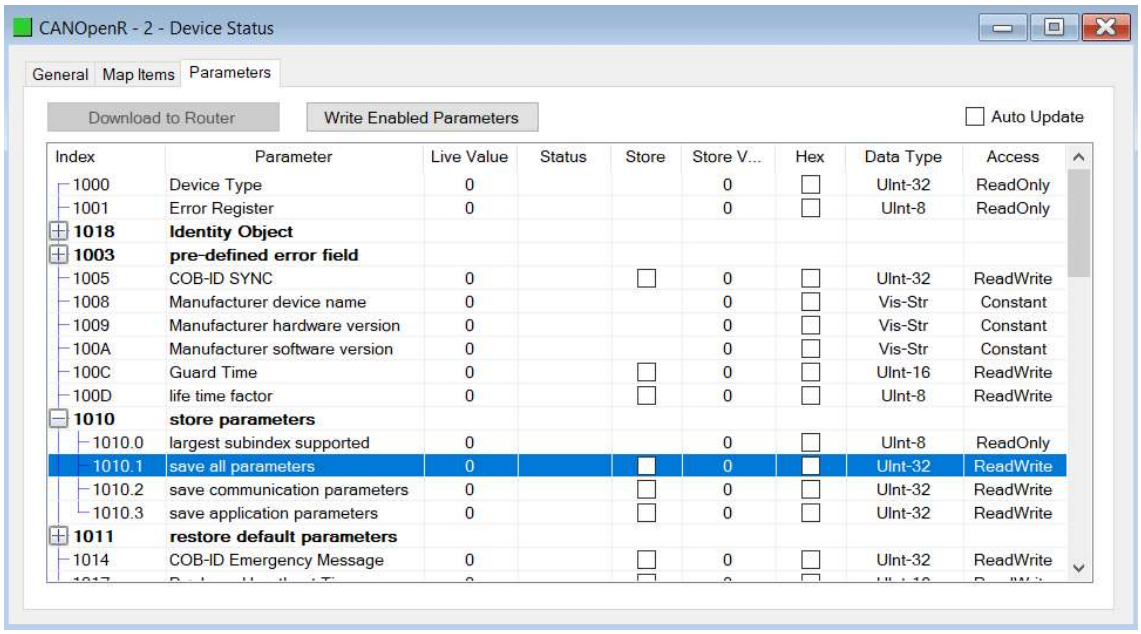


Figure 3.36 – Slave device parameters – Save Parameters

The user can also select to Auto-Update the values in the parameter list by selecting the auto-update checkbox (as shown below). Once set the values will automatically start updating.

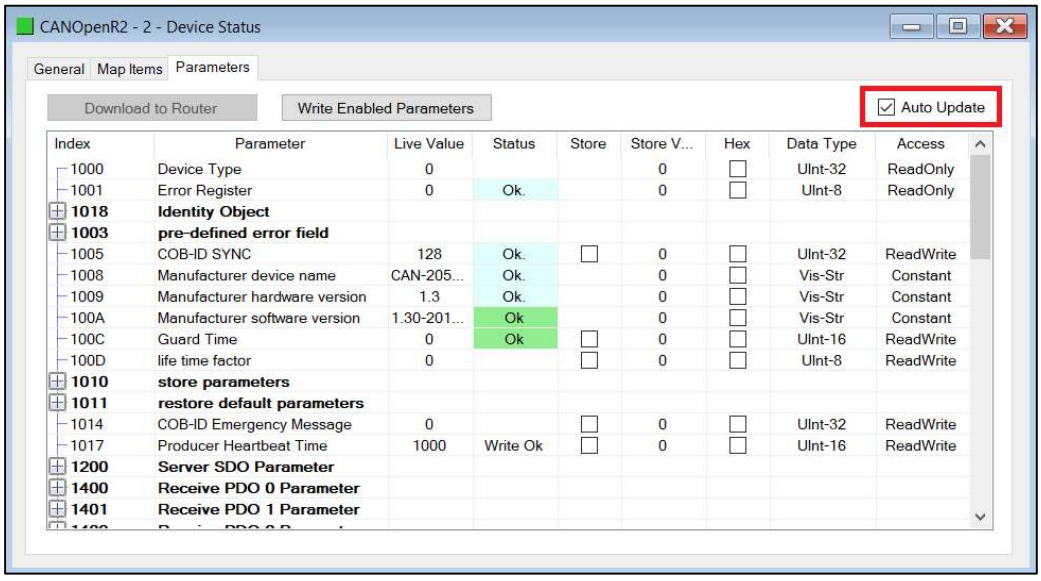


Figure 3.37 – Slave device parameters – Auto Update

3.5.6. 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).

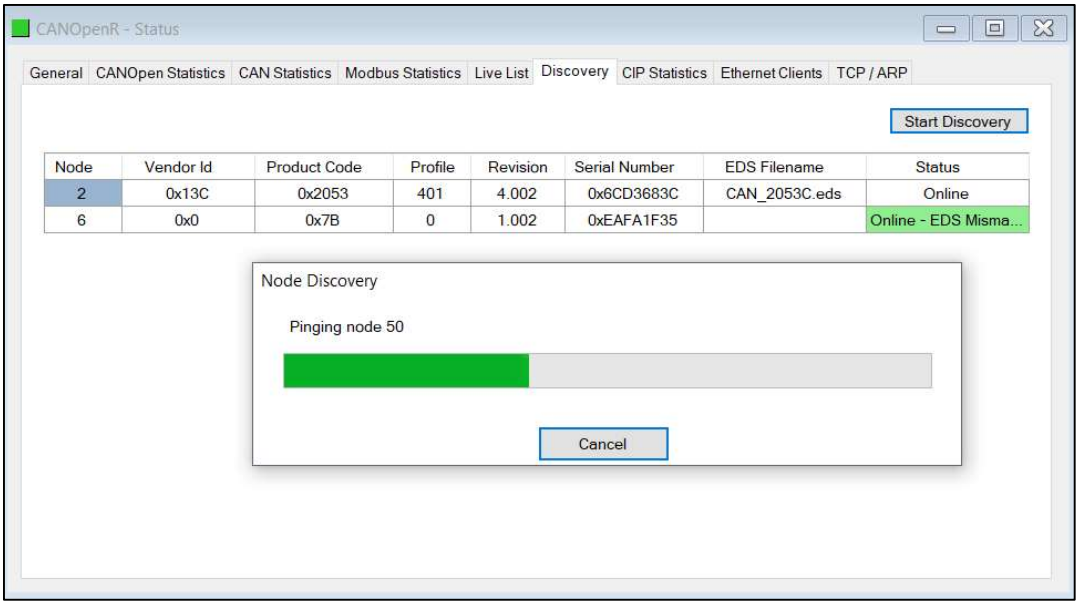


Figure 3.38 – Device Discovery

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).

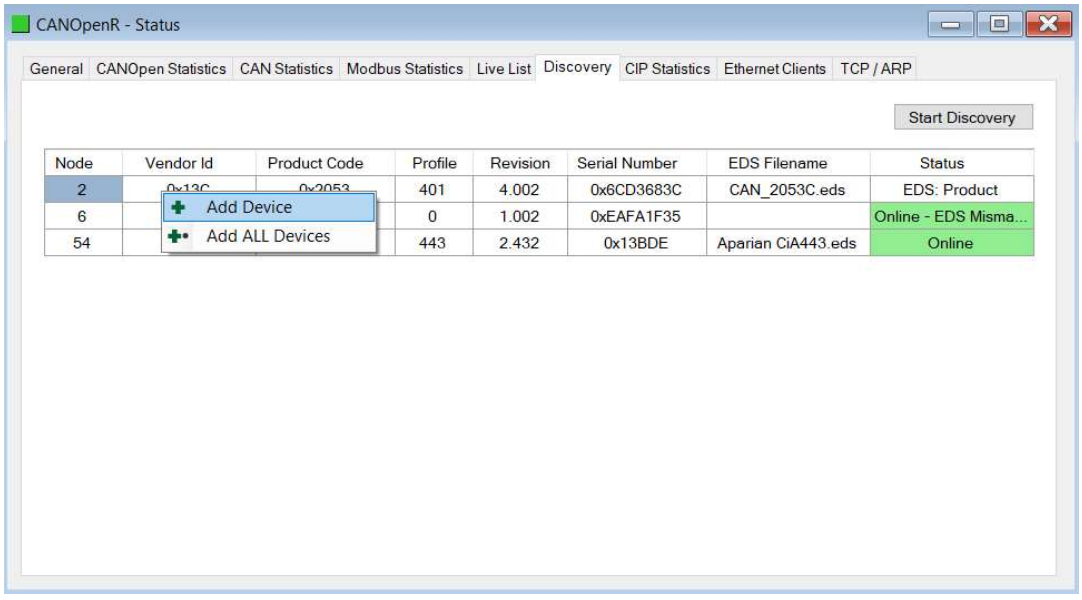


Figure 3.39 – Device Discovery - Add

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 user will be able to map up to 16 PDOs per CANopen Router when operating as a Slave on the CANopen network.

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).

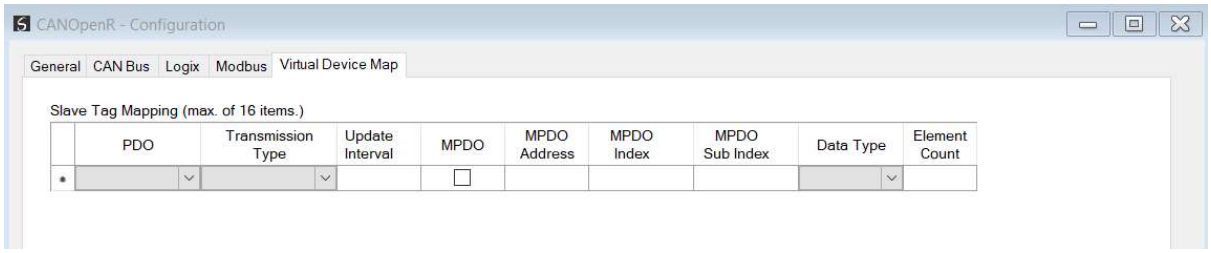


Figure 3.40 – CANopen Router as Slave – PDO Mapping

When the primary interface is EtherNet/IP, the CANopen Router will allow the user to receive data from a CANopen Master and write it into a Logix controller and/or send data to a CANopen Master from a Logix controller.

When the primary interface is Modbus TCP Slave, the CANopen Router will allow the user to received data from a CANopen Slave device and write it into a Modbus Holding Register and/or send data to a CANopen Master from a Modbus Holding Register.

Parameter	Description
Function	There are two functions supported for mapping PDOs (process variables) for the CANopen Router when operating as a CANopen slave. **TPDO x** TPDOs are the PDOs **sent to** the CANopen Master. A total of four TPDOs can be used if multiplexing is not used (see MPDO section). Each PDO received from the Slave device can be up to 8 bytes (e.g. two 32-bit Reals). **RPDO x**

	RPDOs are the PDOs **received from** the CANopen Master. A total of four RPDOs can be used if multiplexing is not used (see MPDO section). Each PDO sent to the Slave device can be up to 8 bytes (e.g. two 32-bit Reals).  **NOTE:** The definitions for the TPDO and RPDO are swapped when the CANopen Router is operating as a CANopen slave. When operating as a CANopen Slave, the TPDO and RPDO are from a field device's perspective.
Transmission Type	**Sync (TPDO only)** The CANopen Router in Slave mode will send out the PDO data to the CANopen Master once a ***SYNC*** packet has been received. **RemoteTxReq (RPDO only)** The CANopen Router will receive the PDO 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-Logix (TPDO only)** The CANopen Router in Slave mode will send out the PDO data to the CANopen Master every time the relevant PDO bit in the ***SlaveModeOutputTriggers*** of the Logix output assembly or Modbus Holding Register.
Update Interval	The time (in milliseconds) at which the PDOs will be sent (when transmission type is ***Evt-Timer***).
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 Logix Tag or Modbus Holding Register.
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 element count multiplied by the data type size must not be greater than 8 bytes when **not** using multiplexing and 4 bytes when using multiplexing.

Target Tag	When the Primary Interface is EtherNet/IP, this parameter will be the Logix Tag that will be used to exchange data with the CANopen Master. The target tag can either be entered manually or if online with the controller the target tag can be updated using the target browser (see figure below).

Table 3.7 –Device Mapping parameters

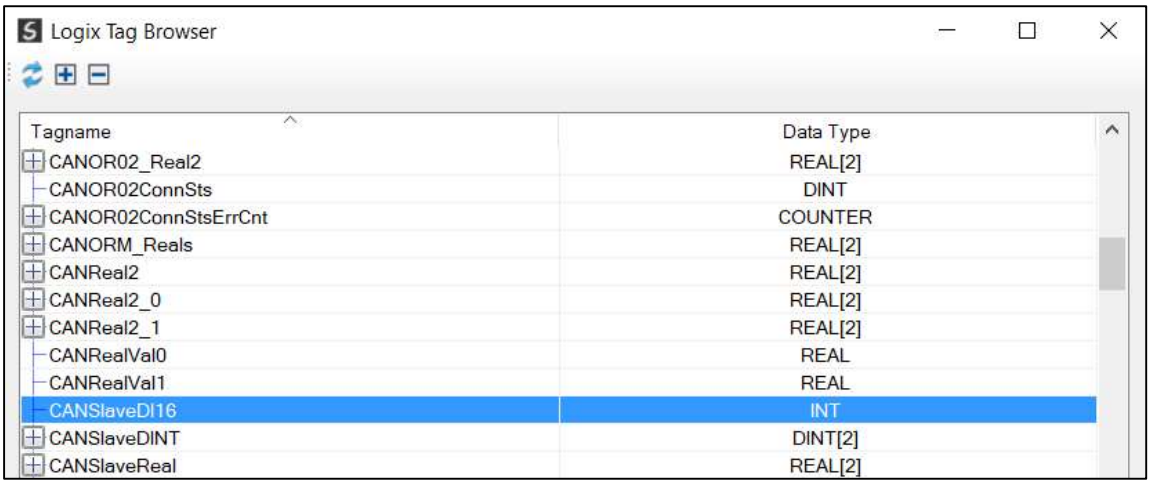


Figure 3.41 – Target Tag selection

3.6.1.1. ETHERNET/IP INTERFACE

When using the EtherNet/IP interface, the RPDO 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.

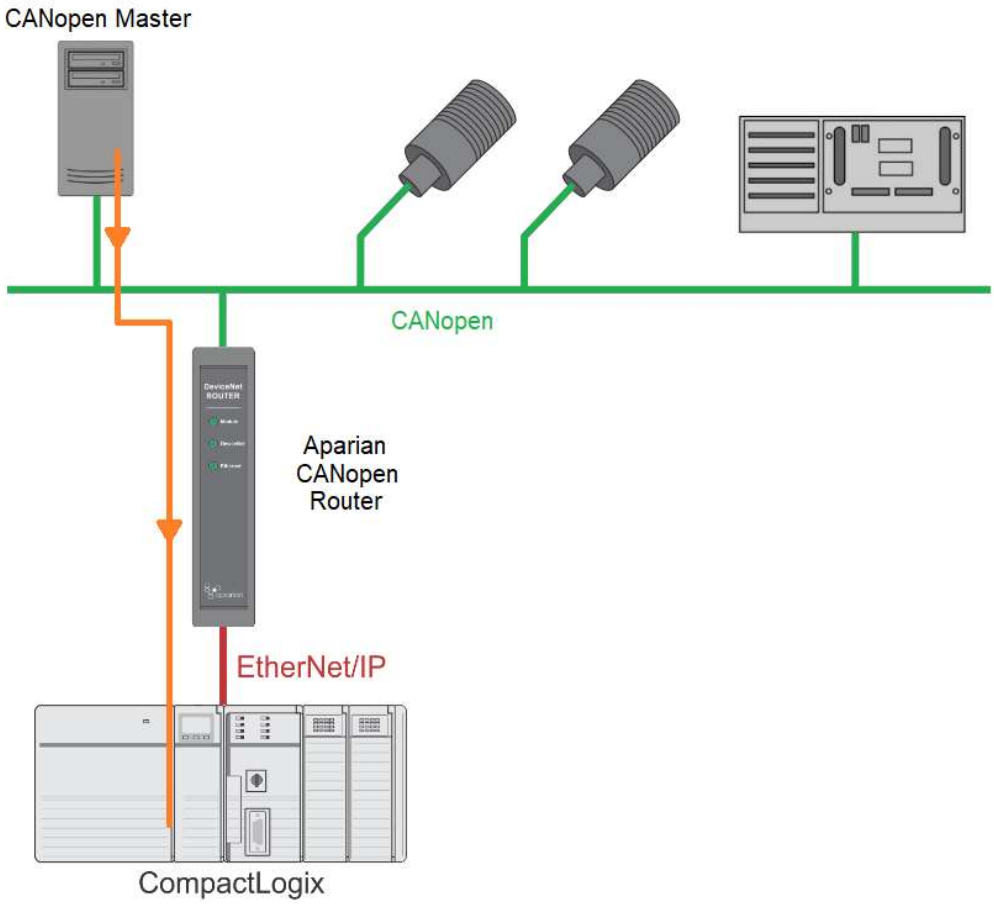


Figure 3.42 – Process variable (RPDO) from CANopen Master to Target Tag

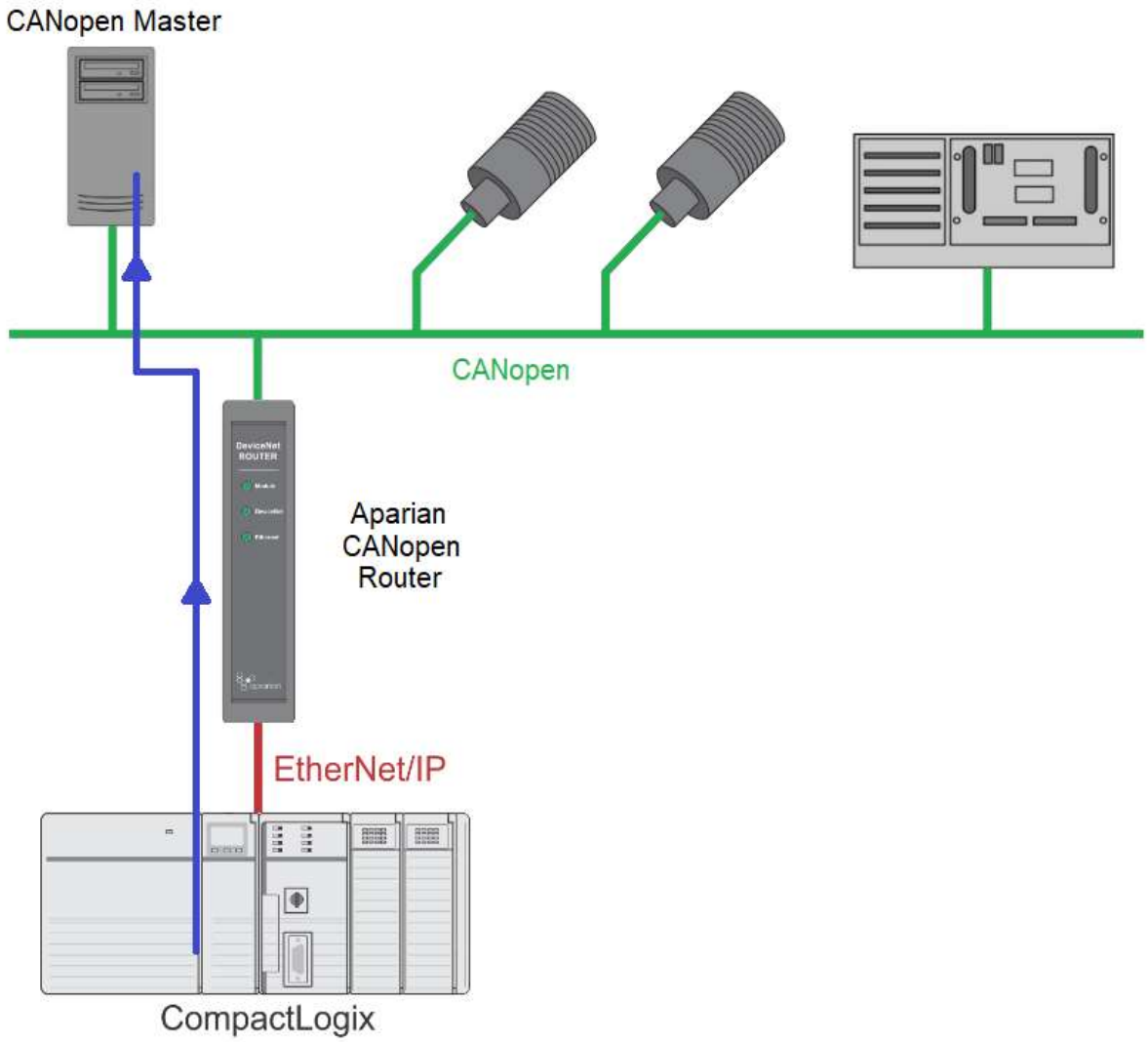
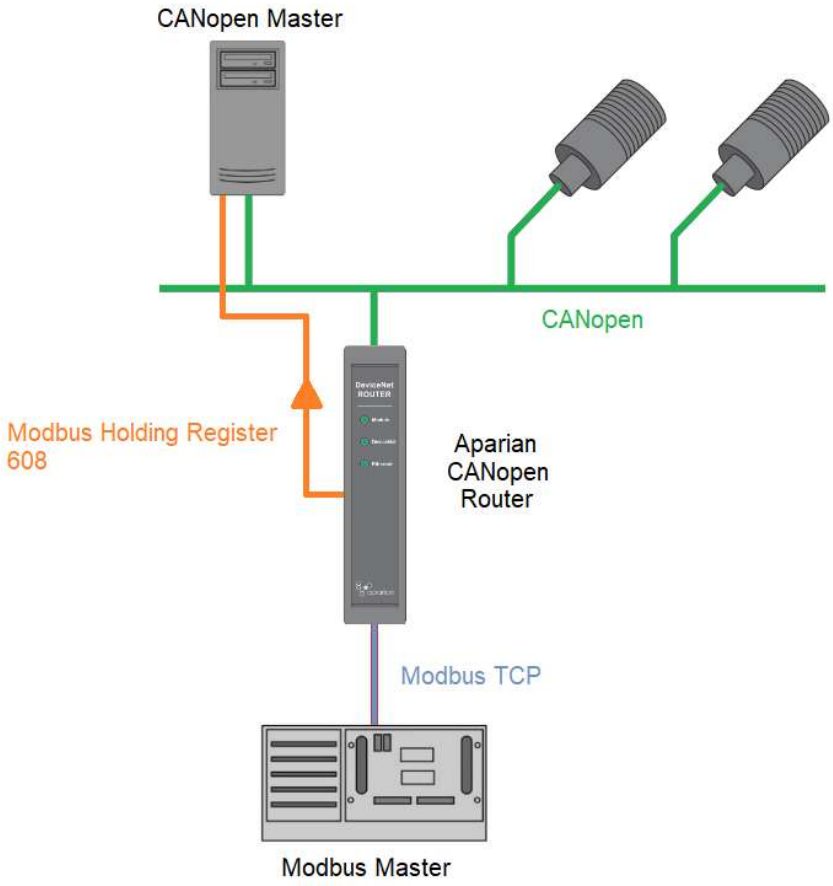


Figure 3.43 – Process variable (TPDO) sent to CANopen Master from Target Tag

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, the Logix Target Tag cannot be only one REAL.

3.6.1.2. MODBUS TCP INTERFACE

When Modbus TCP has been selected as the primary interface, the process variables (TPDOs) from the CANopen Router will be read from a predefined Modbus Holding Register. The process variables (RPDOs) that is received from the CANopen Master will be written to predefined internal Modbus Registers.



General | CAN Bus | Logix | Modbus | Virtual Device Map

Slave Tag Mapping (max. of 16 items.)

	PDO	Transmission Type	Update Interval	MPDO	MPDO Address	MPDO Index	MPDO Sub Index	Data Type	Element Count
	TPDO 1	Evt - Timer	1000	☐				Real-32	1
	TPDO 2	Evt - Timer	1000	☐				Real-32	1
	TPDO 3	Evt - Timer	1000	☐				Real-32	1
*				☐					

Figure 3.44 – Process variable (TPDO) from Modbus Holding Register to CANopen Master

In the above example the PDO value will be written to Modbus Holding Register 608. This is calculated as follow:

Modbus Holding Register
Slave Mode start– *600*
Slave Mode PDO 3 – *600 + ((PDO -1) * 4) = 608*

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

NOTE: The PDO offset in the Modbus Holding Register will depend on its location in the Mapping. For example, if TPDO 3 at mapping index 5 (as shown

below) will be at Modbus Holding Register 616 *(600 + (5 – 1) * 4)*. The offset in the Modbus Holding registers is independent of the TPDO number (e.g. TPDO **2**).

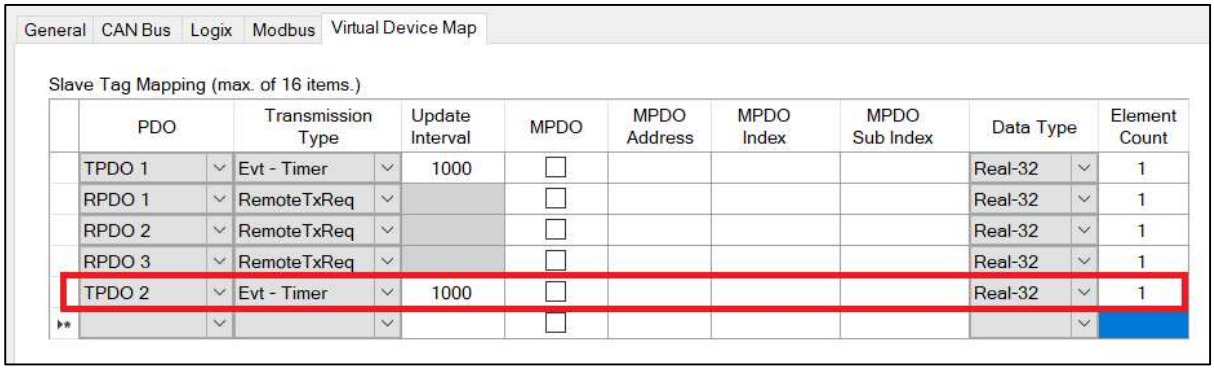


Figure 3.45 – Process variable (TPDO) at mapping index 5

NOTE: When receiving process variables from a CANopen Master, the same example and calculation as above applies.

NOTE: To optimise the Modbus communication it is recommended to group all the TPDOs together and then all the RPDOs.

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..

3.7. MODULE DOWNLOAD

Once the CANopen Router 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.

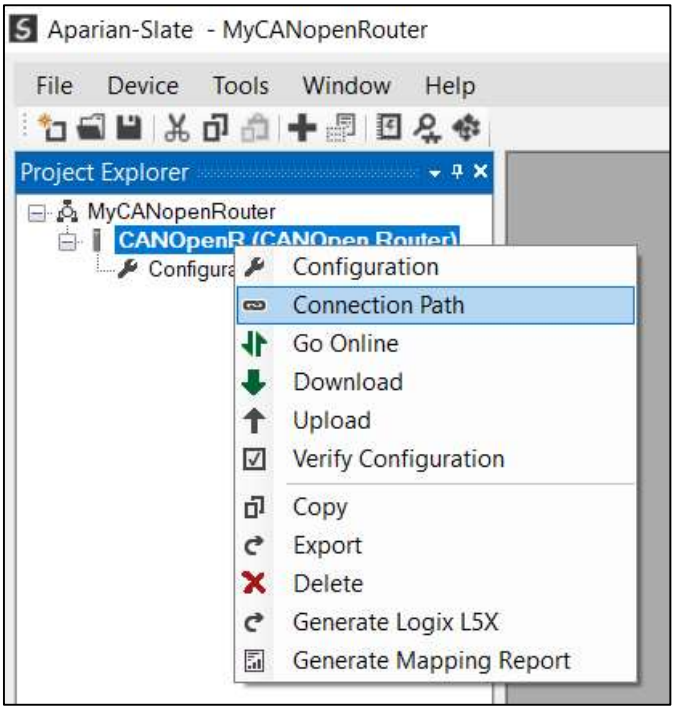


Figure 3.46 - Selecting Connection Path

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

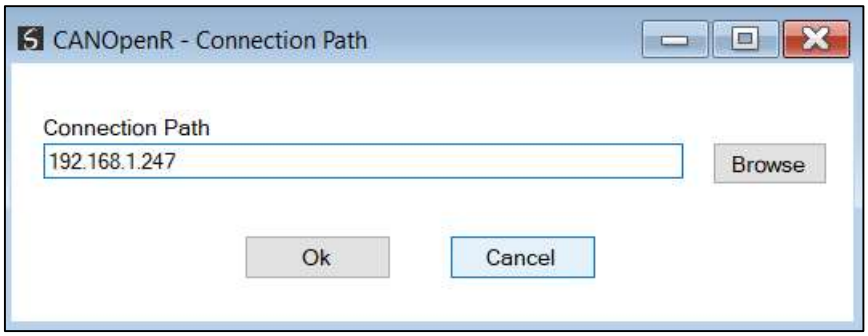


Figure 3.47 - Connection Path

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

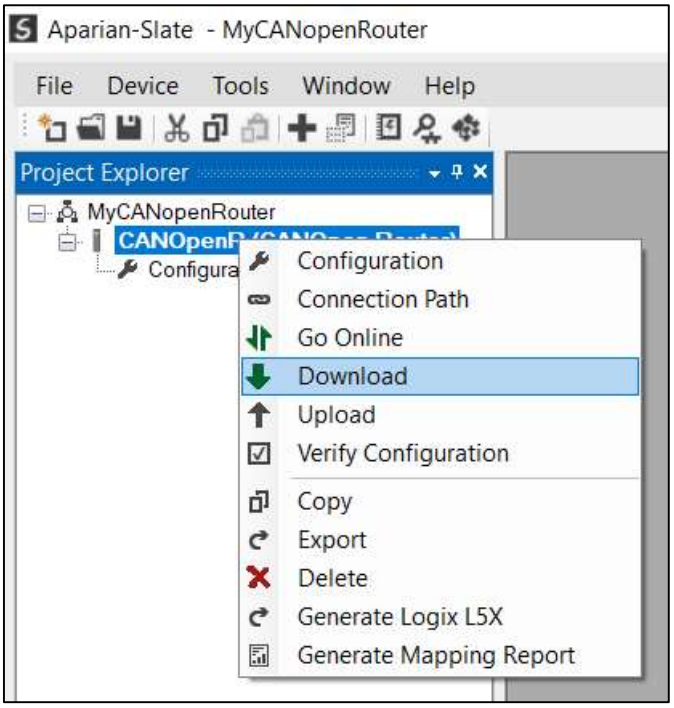


Figure 3.48 - Selecting Download

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

Figure 3.49 - 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.

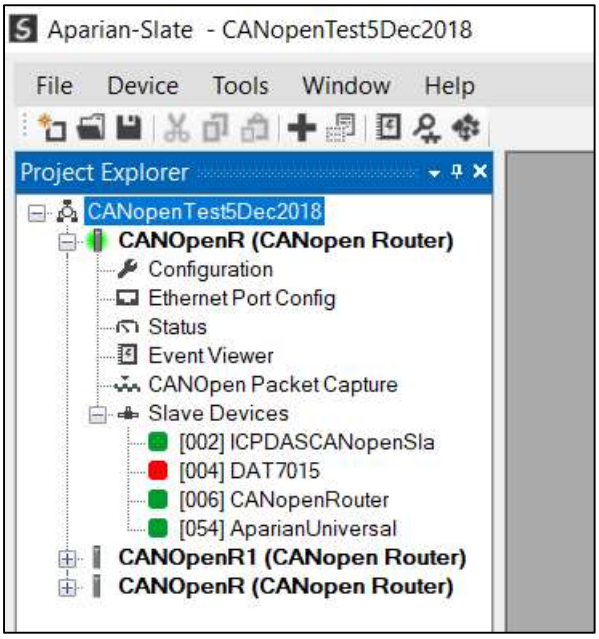


Figure 3.50 - Module online

3.8. LOGIX 5000 CONFIGURATION

3.8.1. ADD MODULE TO I/O CONFIGURATION

When the module operates in a Logix "owned" mode the CANopen Router will need to be added to the Logix 5000 I/O tree. The module will need to be added as a generic Ethernet module. This is done by right clicking on the Ethernet Bridge in the Logix 5000 and selecting *New Module* after which the *ETHERNET-MODULE* is selected to be added as shown in the figure below.

NOTE: See the next section for importing the configuration (L5X).

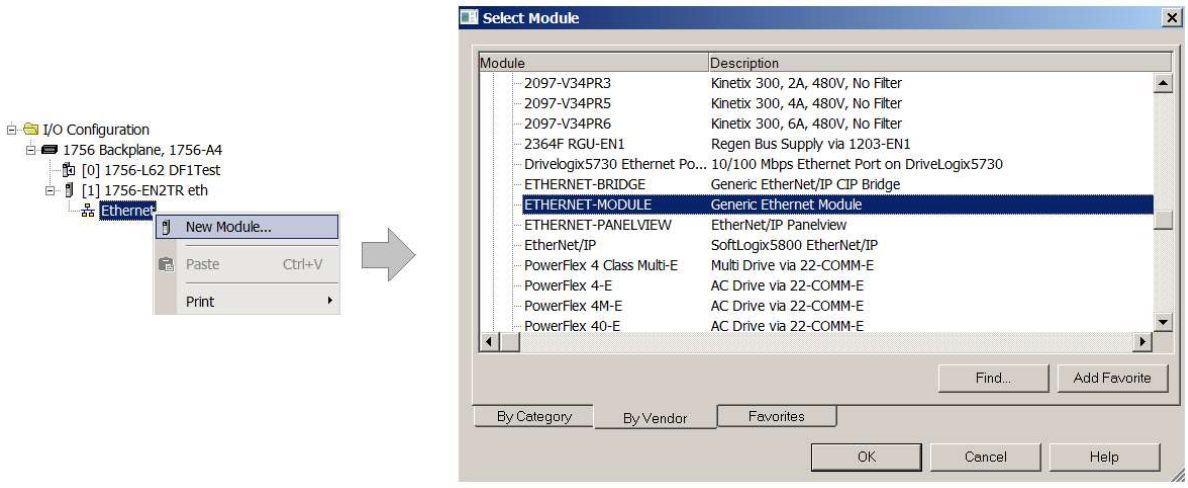


Figure 3.51 - Add a Generic Ethernet Module in Logix 5000

The user must enter the IP address of the CANopen Router that will be used. The assembly instance and size must also be added for the input, output, and configuration in the connection parameters section. Below are the required connection parameters.

Connection Parameter	Assembly Instance	Size
Input	125	91 (32-bit)
Output	126	68 (32-bit)
Configuration	102	0 (8-bit)

Table 3.8 - Logix class 1 connection parameters for the CANopen Router

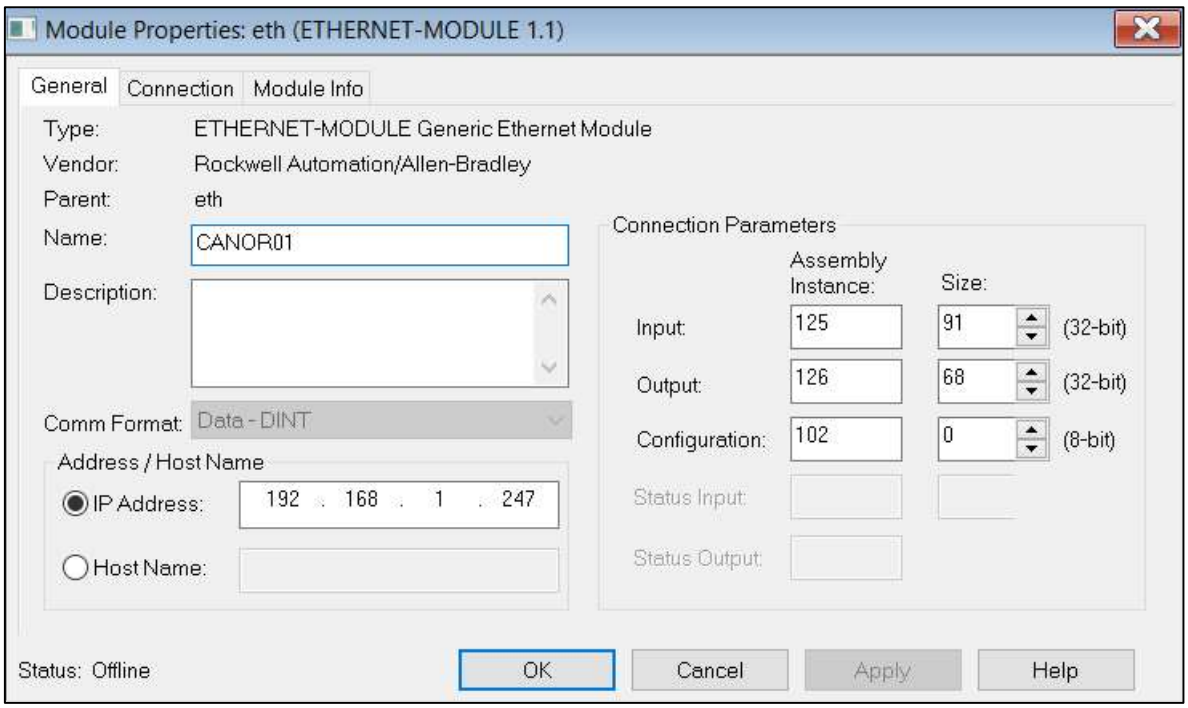


Figure 3.52 - Logix General module properties in Logix 5000

NOTE: The user will need to enter the exact connection parameters before the module will establish a class 1 connection with the Logix controller.

Next the user needs to add the connection requested packet interval (RPI). This is the rate at which the input and output assemblies are exchanged. The recommended value is 100ms. Refer to the technical specification section in this document for further details on the limits of the RPI.

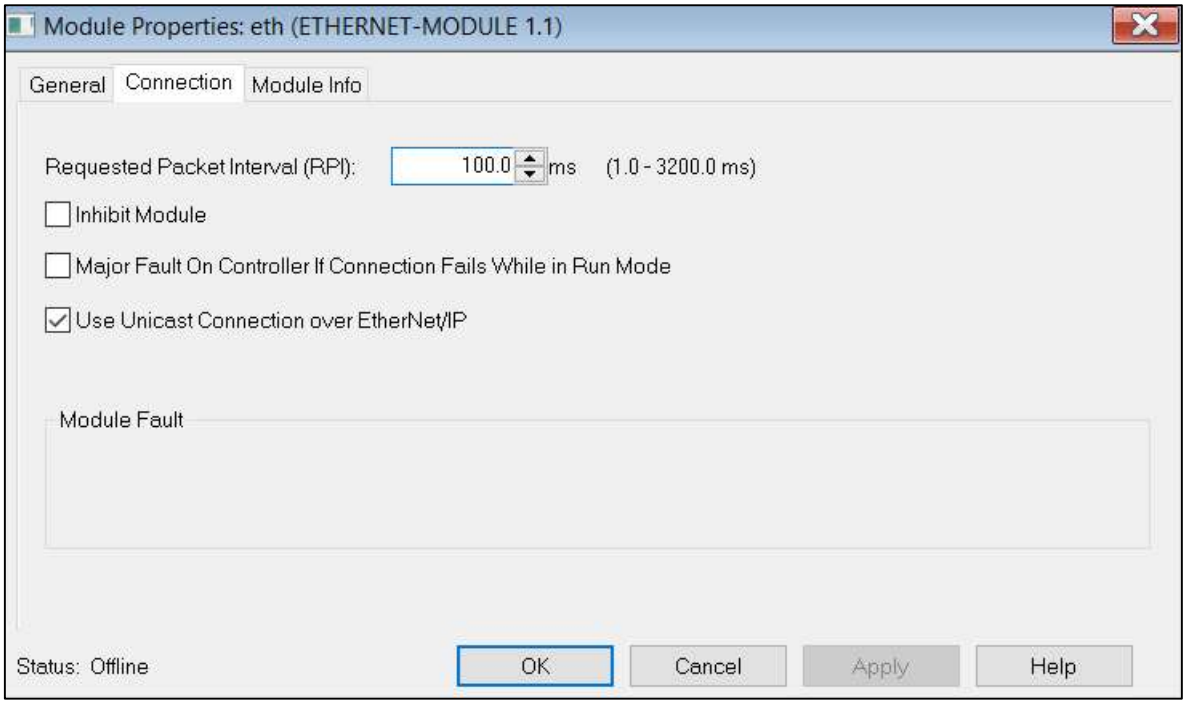


Figure 3.53 - Connection module properties in Logix 5000

Once the module has been added to the Logix 5000 I/O tree the user must assign the User Defined Types (UDTs) to the input and output assemblies. The user can import the required UDTs by right-clicking on *User-Defined* sub-folder in the *Data Types* folder of the IO tree and selecting *Import Data Type*. The assemblies are then assigned to the UDTs with a ladder copy instruction (COP) as shown in the figure below.

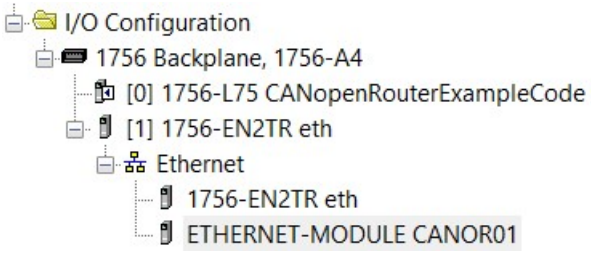


Figure 3.54 – Logix 5000 I/O module tree

3.8.2. 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.

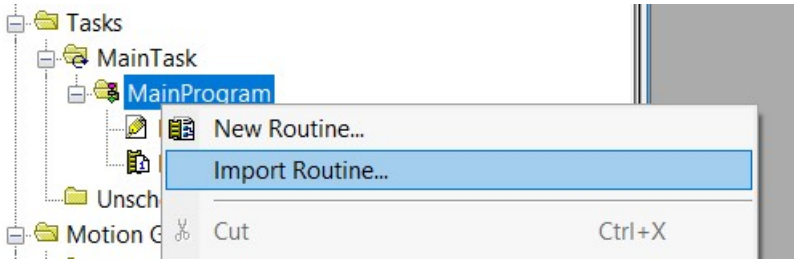


Figure 3.55 – Logix 5000 Importing CANopen Router specific routine and UDTs

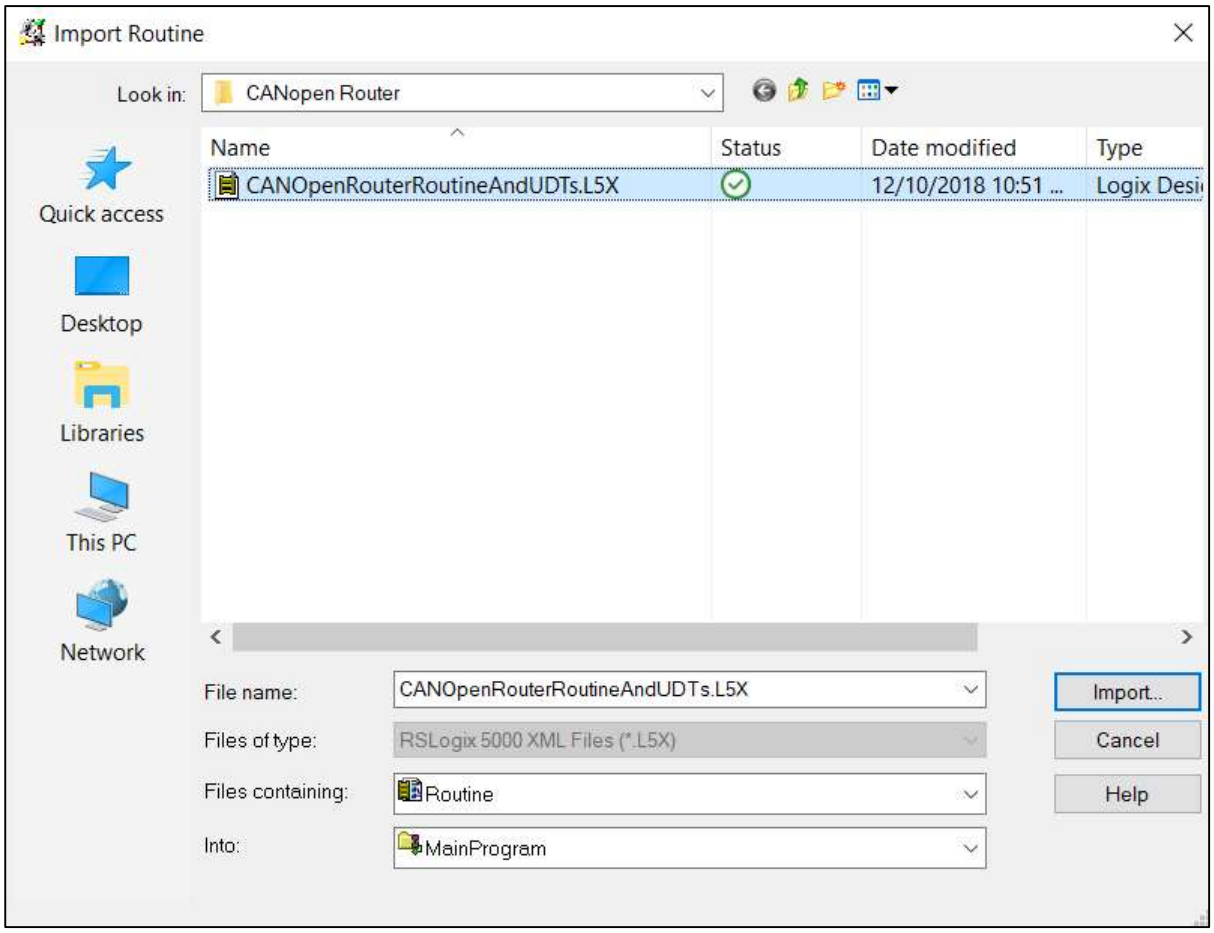


Figure 3.56 - Selecting partial import file

The import will create the following:

- The required UDTs (user defined data types)
- Two controller tags representing the Input and Output assemblies.
- A routine mapping the CANopen Router 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.

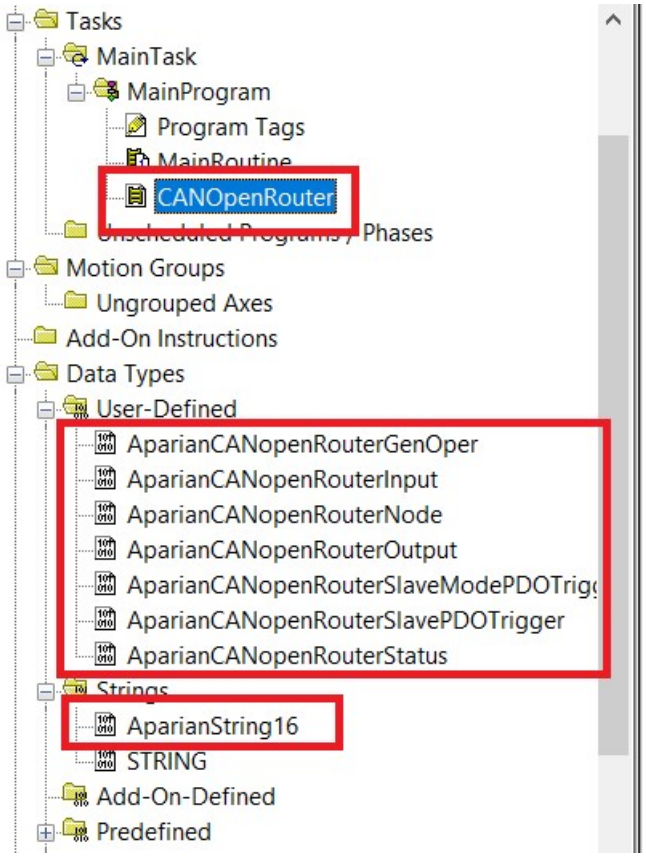


Figure 3.57 - Imported Logix 5000 objects

Refer to the additional information section of this document for an example Logix 5000 project as well as the required UDTs.

4. OPERATION

4.1. LOGIX MESSAGE ROUTING

Once the module has been configured correctly, the CANopen Router will be ready to send and receive the configured PDOs on the CANopen network and route the data to and from the selected Logix tags. Once the PDO is successfully sending or receiving data ***PDOxOk*** bit in the respective Slave input assembly will be set. Refer to the diagnostics section of this document for a more detailed explanation of the various indicators that can be used to diagnose the module.

4.2. LOGIX ASSEMBLIES

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

NOTE: If communication to the Logix controller is lost, then (when in Master mode) the CANopen Router will force the CANopen network to the pre-operational state.

4.2.1. INPUT ASSEMBLY

The following parameters are used in the input assembly 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 configuration in Slate.
InstanceName	SINT[16]	This parameter is the instance name of the module that was configured under the general CANopen Router 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.MBOnline	BOOL	This is reserved for EtherNet/IP interface.
Status.Inhibited	BOOL	Module CANopen sending and receiving has been inhibited.
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.
CrcErrCanCount	DINT	CAN CRC failed message count.
BitErrCanCount	DINT	CAN Bit error count.
StuffErrCanCount	DINT	CAN Stuff error count.
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.
Slave[**x**]	AparianCANopenRouterNode[64]	A total of 64 CANopen slaves can be mapped into the Logix input assembly. The below structure will be repeated for each mapped CANopen Slave. **SlaveAddress** The node address of the mapped slave on the CANopen network. **Online** 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. **ErrorReceived** Set when the last EMCY message received from the slave has an error. **PdoError** Set if one of the PDOs are not operating correctly. **Initializing** Set when the slave is in the initialize state. **Stopped** Set when the slave is in the stopped state. **Operational** Set when the slave is in the operational state. **PreOperational** Set when the slave is in the pre- operational state. **PDOxOk** Each PDO (max 16) has a bit to indicate that it is operating as expected.

Table 4.1 - Logix 5000 input assembly parameters

4.2.2. OUTPUT ASSEMBLY

The following parameters are used in the output assembly 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.
SlaveModeOutputTriggers.TxPDO**x**Trigger	BOOL[16]	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 – Logix**. 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.
SlaveOutputTriggers[**x**].TxPDO**y**Trigger	BOOL[16]	When the CANopen Router is operating as a CANopen Master, these bits are used to trigger **sending** of PDOs to the CANopen Slave device when the Transmission Type is **Evt – Logix**. 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 Slave device.

Table 4.2 - Logix 5000 output assembly parameters

4.3. CIP MESSAGING

The CANopen Router will allow the user to extract certain information from CANopen Slave devices using CIP messages. Below are the required parameters for SDO parameter extraction from the slave device as well as operational data from the slave device.

4.3.1. SDO PASSTHROUGH

4.3.1.1. CIP MESSAGE:

Parameter	Description
Service Code	0x65 (Hex)
Class	0x417 (Hex)
Instance	1
Attribute	N/A

Request Data Length	9 - 489

Table 4.3 – SDO Passthrough Message

4.3.1.2. 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 4.4 – SDO Passthrough Request

4.3.1.3. 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 4.5 – SDO Passthrough Response

4.3.2. SLAVE INFORMATION

4.3.2.1. CIP MESSAGE:

Parameter	Description

Service Code	0x66 (Hex)
Class	0x417 (Hex)
Instance	1
Attribute	N/A
Request Data Length	2

Table 4.6 – Slave Information Message

4.3.2.2. REQUEST DATA:

Parameter	Data Type	Description
Node	SINT	The Node Address of the CANopen Slave
Command	SINT	The command to be sent to the CANopen Slave device. 0 – Return the status of the Slave. 1 – Reset Node 2 – Set Mode to Operational 3 – Set Mode to Pre-Operational (PDOs will stop communicating) 4 – Set Mode to Operational (PDOs and SDOs will stop communicating)

Table 4.7 – Slave Information Request

4.3.2.3. RESPONSE DATA (WHEN COMMAND 0 WAS REQUESTED):

Parameter	Data Type	Description
Status	SINT	Bit 0 – Online Bit 1 – Error Received Bit 2 to 7 - Reserved
State	SINT	0 – Initializing 1 – Disconnected 2 – Connecting 3 – Preparing 4 – Stopped 5 – Operational 126 – Unknown 127 – Pre-operational
Inactive Time	INT	Time since the last communication received from the slave.
Last Error Code	INT	The code of the last error received (refer to the Slave device user manual for a description of the error codes).
Last Error Type	SINT	The type of the last error received. Bit 0 - Generic Bit 1 - Current

		Bit 2 - Voltage Bit 3 - Temperature Bit 4 - Communication Bit 5 - Device Profile Specific Bit 6 - Reserved Bit 7 - Manufacturer Specific
PDO		**Repeat the below for each PDO (max 16)**
Flags	SINT	Flags Bit 0 - Configured Flags Bit 1 - TransactionOk Flags Bit 2 - Transmit PDO Flags Bit 3 - Receive PDO Flags Bit 4 to 7 - Reserved
Transaction Count	DINT	The number of transactions completed.
Timeout	INT	The number of times the PDO receiving has timed out.

Table 4.8 – Slave Information Response

4.4. MODBUS MAPPING

When the primary interface of the CANopen Router is set to Modbus TCP, the CANopen Router will operate as a Modbus TCP Slave supporting the following Modbus registers.

NOTE: If communication to the Modbus Master is lost, then (when in Master mode) the CANopen Router will force the CANopen network to the pre-operational state.

Register Type:	**Holding Registers**			
Parameter	Byte Length	Date Type	Register	Description
Master Status				
Instance Name Length	4	DINT	**0**	This parameter is the instance name length of the module that was configured under the general CANopen Router configuration in Slate.
Instance Name	16	SINT[16]	**2**	This parameter is the instance name of the module that was configured under the general CANopen Router configuration in Slate.
Status *Bit 0 – Configuration Valid* *Bit 1 – Duplicate Node* *Bit 2 – Network Operational* *Bit 3 – Network Pre-Operational* *Bit 4 – Network Stopped*	4	DINT	**10**	

Bit 5 – Master Mode *Bit 6 – Slave Mode* *Bit 7 – Modbus Online* *Bit 8 - Inhibited* *Bit 9 to 31 – Reserved*				
Transaction Rate	4	DINT	**12**	The transaction rate is the number of CANopen messages per second that the module is currently routing.
Device Temperature	4	REAL	**14**	The internal temperature of the CANopen Router module.
UTC Time	8	DINT[2]	**16**	The UTC time on the CANopen network.
RxCANCount	4	DINT	**20**	Received CAN message count.
TxCANCount	4	DINT	**22**	Transmitted CAN message count.
CrcErrCanCount	4	DINT	**24**	CAN CRC failed message count.
BitErrCanCount	4	DINT	**26**	CAN Bit error count.
StuffErrCanCount	4	DINT	**28**	CAN Stuff error count.
PdoTxCount	4	DINT	**30**	The number of PDO packets transmitted.
PdoRxCount	4	DINT	**32**	The number of PDO packets received.
SdoTxCount	4	DINT	**34**	The number of SDO packets transmitted.
SdoRxCount	4	DINT	**36**	The number of SDO packets received.
TimePcktCount	4	DINT	**38**	The number of TIME packets received or sent.
SyncPcktCount	4	DINT	**40**	The number of SYNC packets received or sent.
EmergencyPcktCount	4	DINT	**42**	The number of EMCY packets received or sent.
HeartbeatPcktCount	4	DINT	**44**	The number of Heartbeat packets received.
TagReads	4	DINT	**46**	The total number of Logix tag reads executed by the module.
TagWrites	4	DINT	**48**	The total number of Logix tag writes executed by the module.
ConnectionFailures	4	DINT	**50**	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	4	DINT	**52**	The number of failed tag access (read/write) requests. These may include privileged violations, non-existing tags, etc.
Slave Device Status x 64				
Slave Address	1	SINT	**100 + (2 x Slave Idx)**	The node address of the mapped slave on the CANopen network.

Status *Bit 0 – Online* *Bit 1 – Error Received* *Bit 2 – PDO Error* *Bit 3 – Initializing* *Bit 4 – Stopped* *Bit 5 – Operational* *Bit 6 – Pre-Operational* *Bit 7 – Reserved*	1	SINT	**100 + (2 x Slave Idx)**	**Online** 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. **ErrorReceived** Set when the last EMCY message received from the slave has an error. **PdoError** Set if one of the PDOs are not operating correctly. **Initializing** Set when the slave is in the initialize state. **Stopped** Set when the slave is in the stopped state. **Operational** Set when the slave is in the operational state. **PreOperational** Set when the slave is in the pre-operational state.
PDO Status	2	DINT	**101 + (2 x Slave Idx)**	**PDOxOk** Each PDO (max 16) has a bit to indicate that it is operating as expected.
Master Output				
General Operation *Bit 0 – Pre-Operational* *Bit 1 – Stopped* *Bit 2 – Inhibit* *Bit 3 to 31 – Reserved*	4	DINT	**300**	When the CANopen Router is the CANopen Master, the pre-operational bit will force the CANopen network to be PreOperational. When the CANopen Router is the CANopen Master, this stopped bit will force the CANopen network to be Stopped. When Inhibit has been set it will stop all CANopen communication. **NOTE:** When other NetworkPreOperational and NetworkStop bits are **not** set then the CANopen Router will set the network state to Operational.
UTC	8	DINT[2]	**302**	When the CANopen Router is a CANopen Master, the user can write the UTC Time (Unix Time format in microseconds) which will be converted into the CANopen time

				format for when sending TIME messages.
SlaveModeOutputTriggers *Bit 0 – TxPDO0Trigger* *Bit 1 – TxPDO1Trigger* *…* *Bit 15 – TxPDO15Trigger*	4	DINT	**306**	**TxPDOxTrigger** 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 – Logix**. 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.
Slave Device TPDO Trigger				
SlaveOutputTriggers (x 64 slaves) *Bit 0 – TxPDO0Trigger* *Bit 1 – TxPDO1Trigger* *…* *Bit 15 – TxPDO15Trigger*	4 x 64	DINT	**400 + (2 x PDO Idx)**	**TxPDOyTrigger** When the CANopen Router is operating as a CANopen Master, these bits are used to trigger **sending** of PDOs to the CANopen Slave device when the Transmission Type is **Evt – Logix**. 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 Slave device.
Slave Mode PDOs x 16 PDOs				
PDO x	8 x 16	SINT	**600**	When the CANopen Router is operating as a CANopen Slave, each of the PDOs configured in the Virtual Device Map will be accessible from these Modbus Holding Registers (HR). For example, the first PDO in the Slave Tag Mapping will be at HR 600, PDO number two will be at HR 604, and so on.
Slave Device x 16 PDOs				
Slave Map Index 0 PDO x	8 x 16	SINT	**1000**	When the CANopen Router is operating as a CANopen Master, each of the PDOs configured in the Slave Device Map will be accessible from these Modbus Holding Registers (HR). For example, the first PDO in the Tag Mapping will be at HR 1000, PDO number two will be at HR 1004, and so on.
Slave Map Index 1 PDO x	8 x 16	SINT	**1100**	When the CANopen Router is operating as a CANopen Master, each of the PDOs configured in the Slave Device Map will be accessible from these Modbus Holding Registers (HR).

				For example, the first PDO in the Tag Mapping will be at HR 1100, PDO number two will be at HR 1104, and so on.
Slave Map Index 2 PDO x	8 x 16	SINT	**1200**	When the CANopen Router is operating as a CANopen Master, each of the PDOs configured in the Slave Device Map will be accessible from these Modbus Holding Registers (HR). For example, the first PDO in the Tag Mapping will be at HR 1200, PDO number two will be at HR 1204, and so on.
…	…	…	…	…
Slave Map Index 63 PDO x	8 x 16	SINT	**1000**	When the CANopen Router is operating as a CANopen Master, each of the PDOs configured in the Slave Device Map will be accessible from these Modbus Holding Registers (HR). For example, the first PDO in the Tag Mapping will be at HR 7300, PDO number two will be at HR 7304, and so on.

Table 4.9 – Modbus Mapping

5. DIAGNOSTICS

5.1. LEDS

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

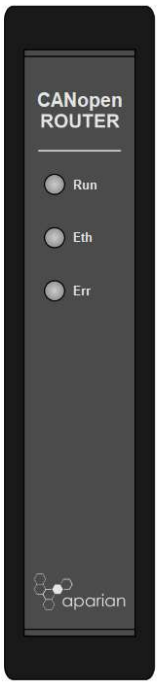


Figure 5.1 - CANopen Router front view

LED	Description
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
Eth	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 was detected.
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 TCP) to the CANopen Router has been lost. **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 TCP) to the CANopen Router has been lost. **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.

Table 5.1 - Module LED operation

5.2. MODULE STATUS MONITORING IN SLATE

The CANopen Router 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.

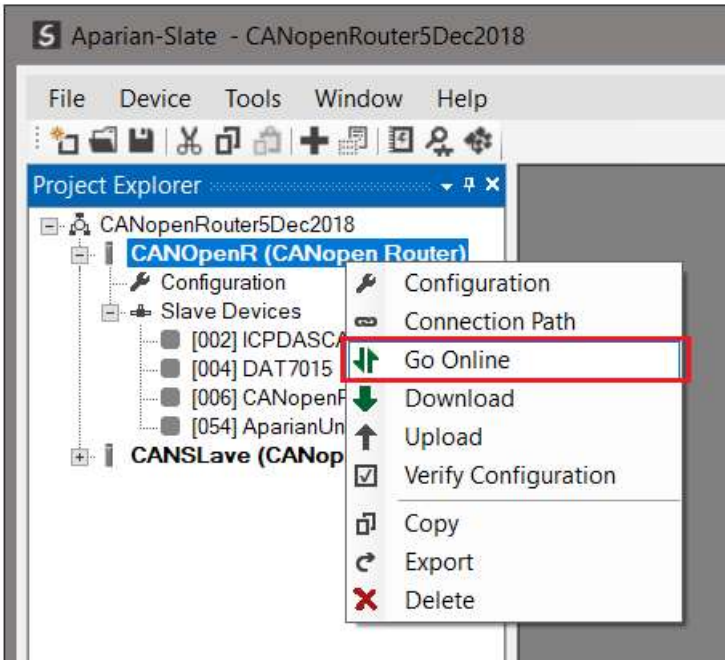


Figure 5.2. - Selecting to Go Online

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

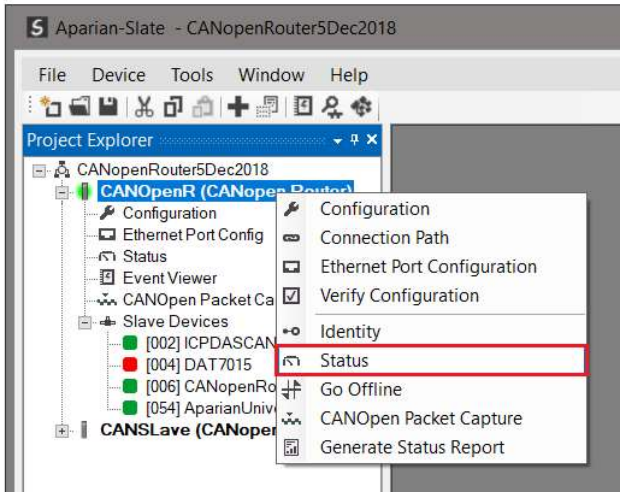


Figure 5.3. - Selecting online Status

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.

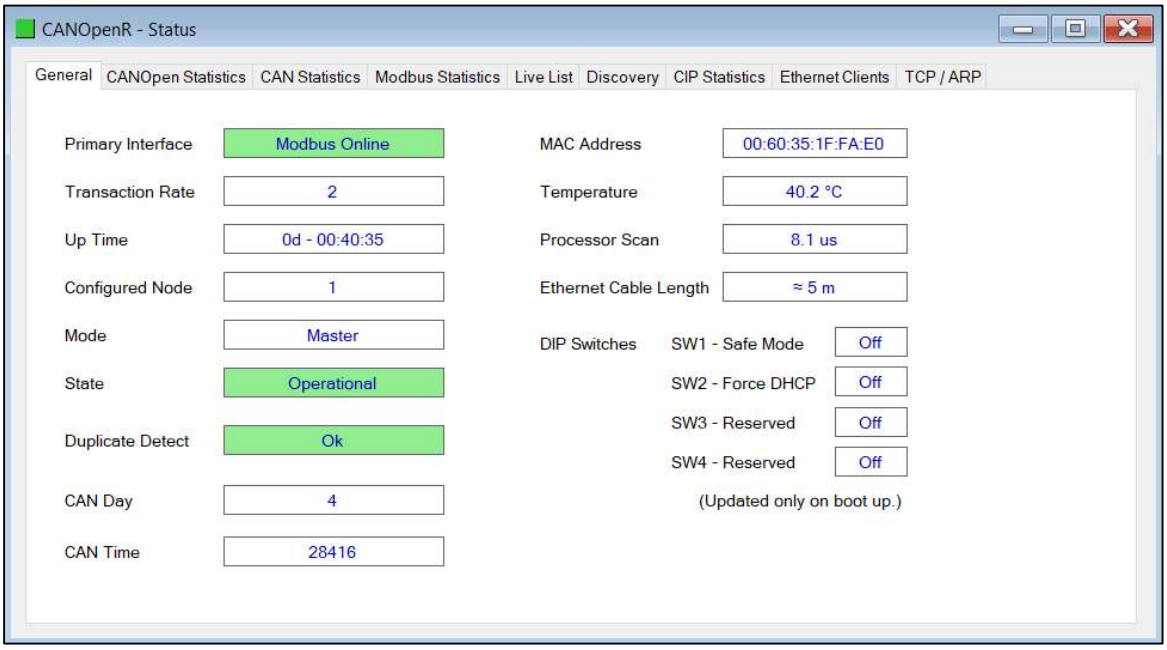


Figure 5.4. - Status monitoring - General

The General tab displays the following general parameters and can also be used to set the module time to the PC time:

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.
Up Time	Indicates the elapsed time since the module was powered-up.
Configured Node	The user required node address as specified in the module configuration.
Mode	The CANopen Router can either be configured as a CANopen Master or CANopen Slave.
State	The operational state of the CANopen network. **Operational** **Pre-operational** **Stopped**
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.
Ethernet Cable Length	Approximate length of the Ethernet cable (accurate to 5m).
DIP Switch Position	The status of the DIP switches when the module booted. Note that this status will not change if the DIP switches are altered when the module is running.

Table 5.2 - Parameters displayed in the Status Monitoring – General Tab

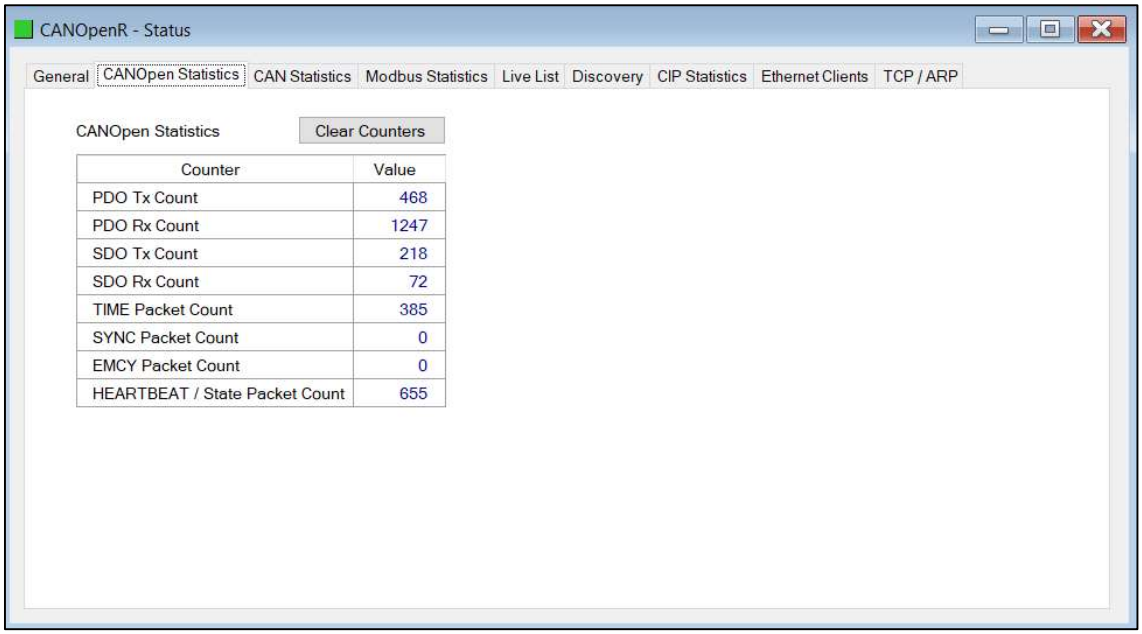


Figure 5.5. - Status monitoring – CANopen Statistics

The CANopen Statistics tab displays the following general 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 5.3 - Parameters displayed in the Status Monitoring – CANopen Statistics Tab

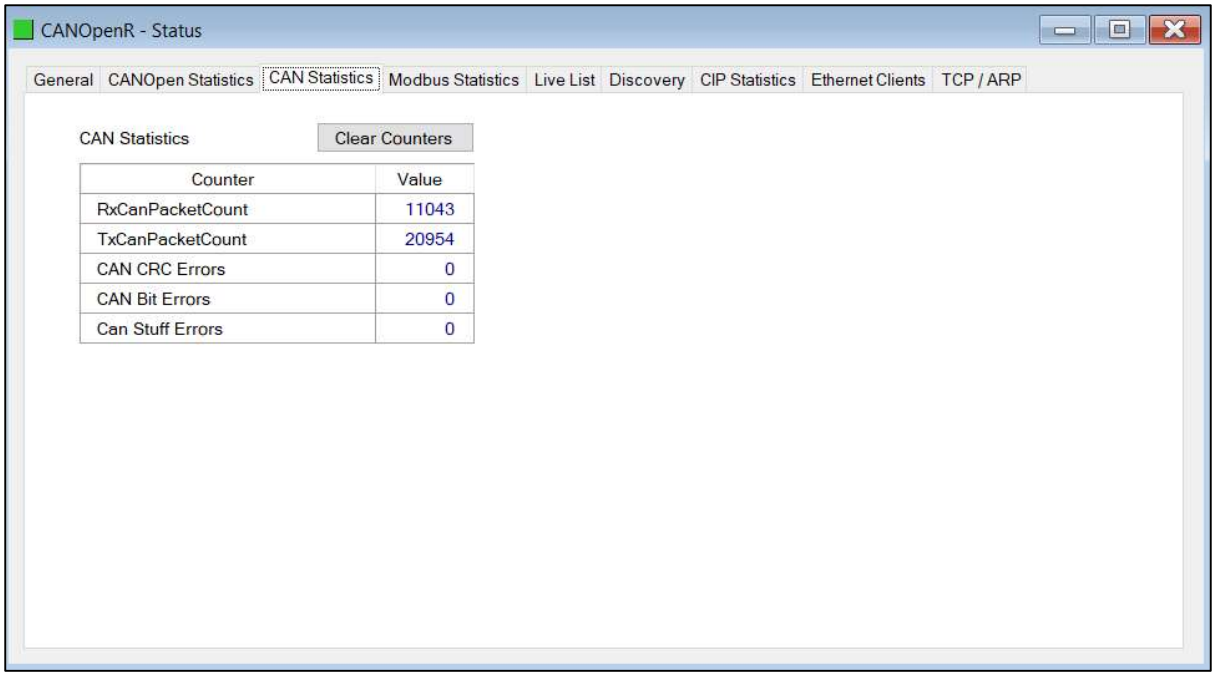


Figure 5.6. - Status monitoring – CAN Statistics

The CAN Statistics tab displays the following general parameters:

Parameter	Description
RxCANPacketCount	Received CAN message count.
TxCANPacketCount	Transmitted CAN message count.
CAN CRC Errors	CAN CRC failed message count.
CAN Bit Errors	CAN Bit error count.
CAN Stuff Errors	CAN Stuff error count.

Table 5.4 - Parameters displayed in the Status Monitoring – CAN Statistics Tab

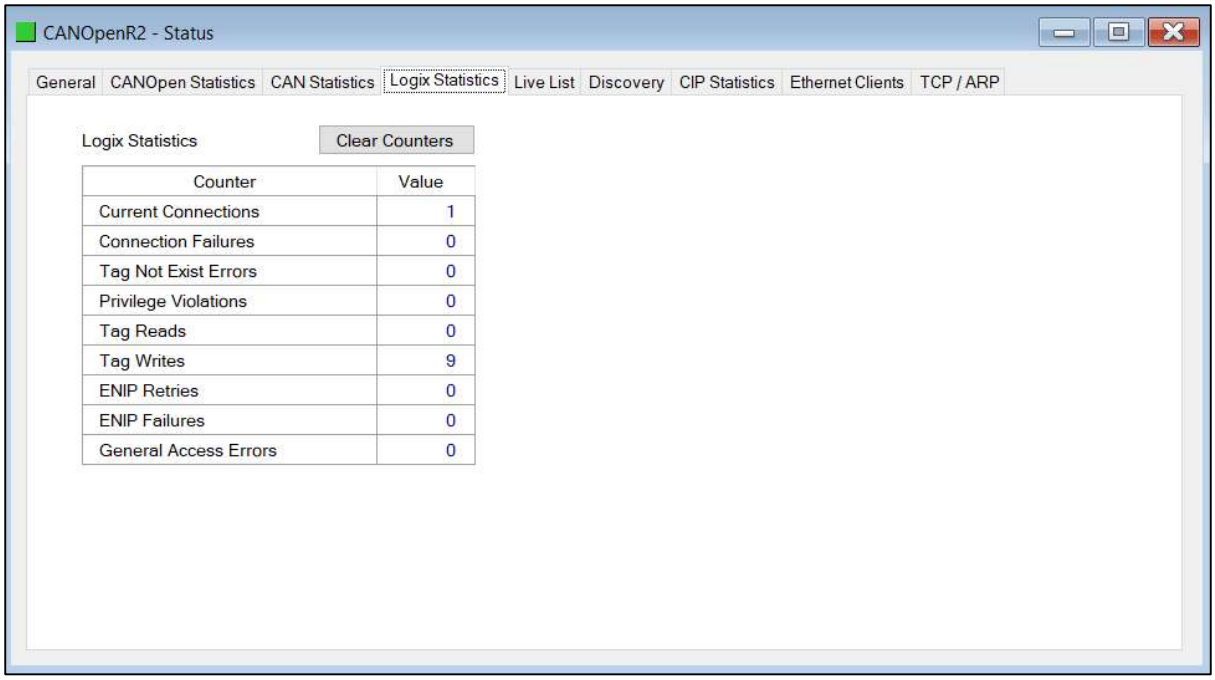


Figure 5.7. - Status monitoring – Logix Statistics

The Logix Statistics tab displays the following general 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 module.
Tag Writes	The number of tag write transactions executed by the CANopen Router 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 5.5 - Parameters displayed in the Status Monitoring – Logix Statistics Tab

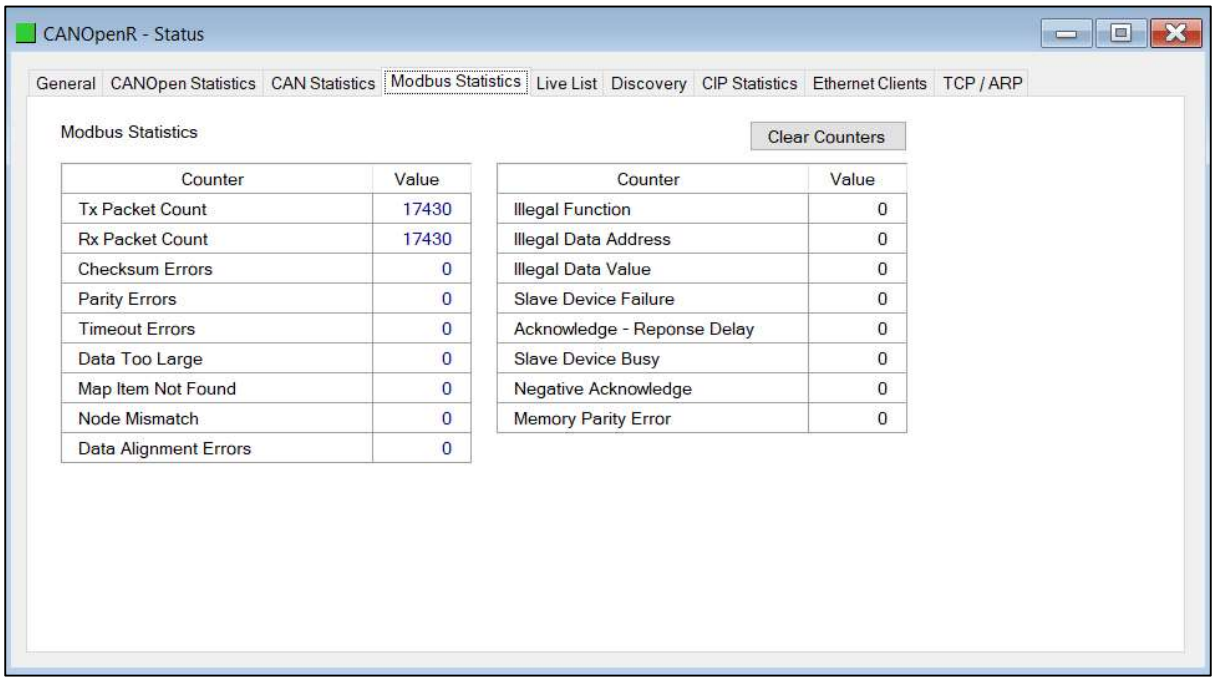


Figure 5.5. - Status monitoring – Modbus Statistics

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 5.6 - Parameters displayed in the Status Monitoring – Modbus Statistics Tab

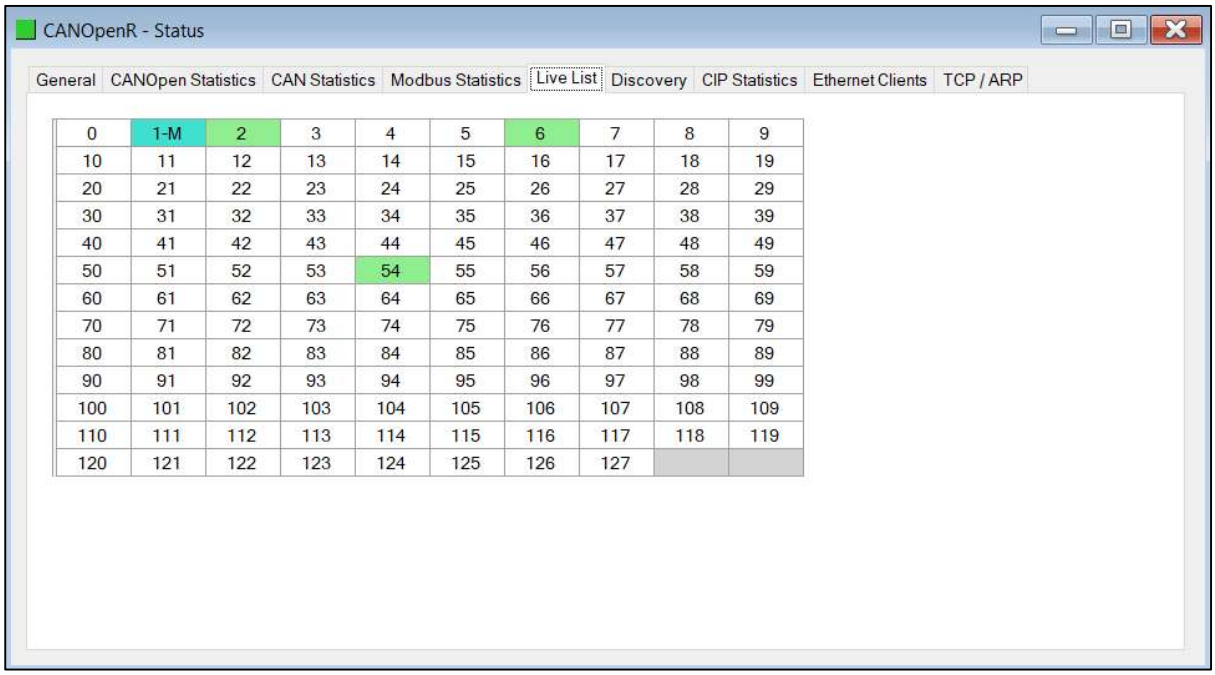


Figure 5.6. - Status monitoring – Live List

The Live List provides the online status of each CANopen Slave device on the CANopen network. When the address in the live list is 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). When the address in the live list is turquoise with "-M" next to it then it is the node address of the CANopen Master.

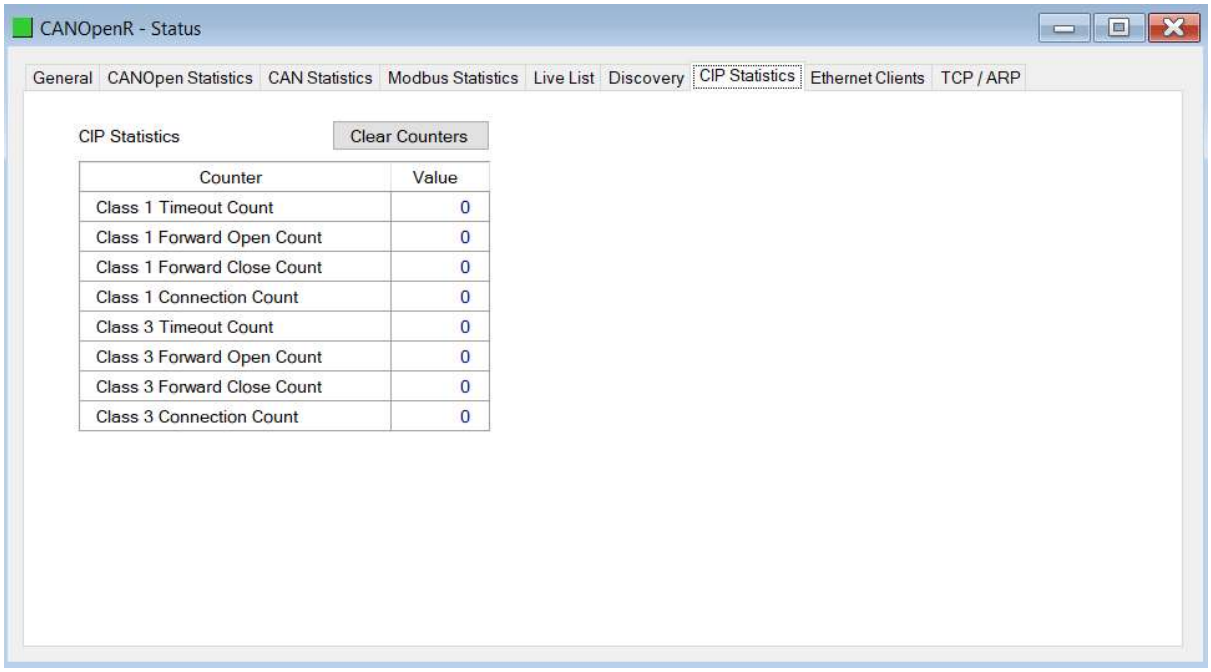


Figure 5.7. - Status monitoring – CIP statistics

The CIP statistics tab displays the following general 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 5.7. - CIP Statistics

5.3. 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.

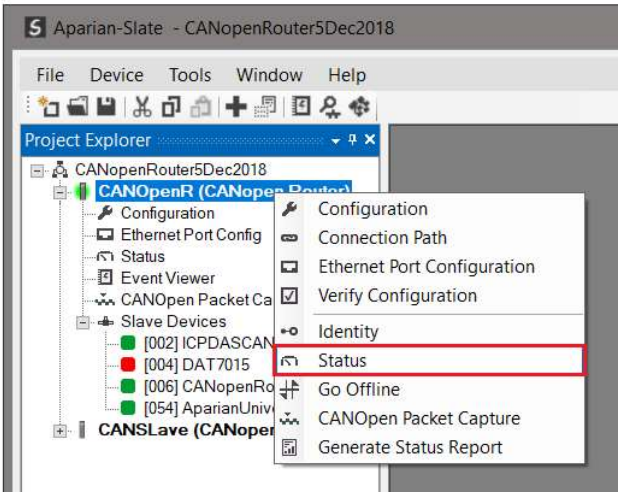


Figure 5.8. - Selecting to Go Online

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

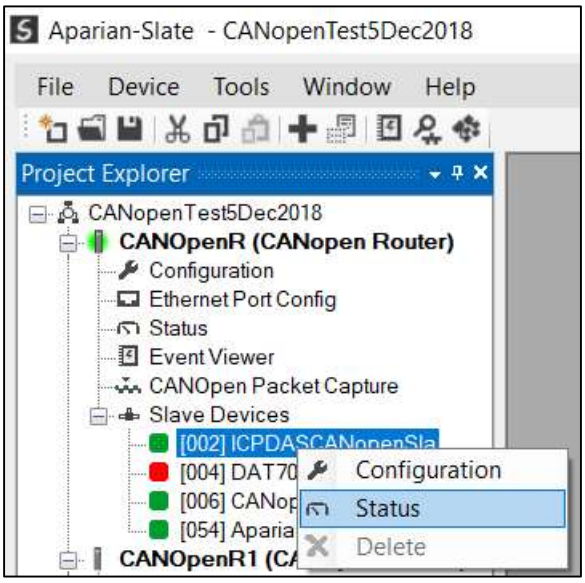


Figure 5.9. - Selecting Slave Device Online Status

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

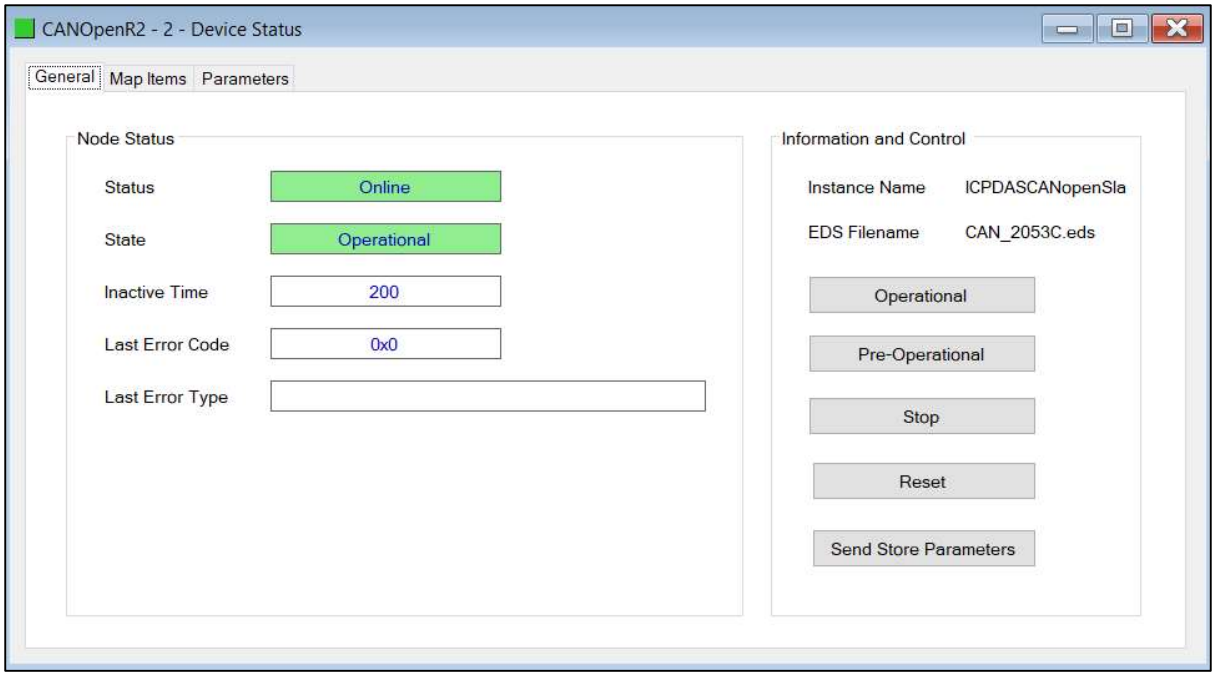


Figure 5.10. – CANopen Slave Device Online Status

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

Statistic	Description
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.
Last Error Code	The last error code received from the slave device.
Last Error Type	The last error type received from the slave device.
Control	The status form allows the user to send certain control messages to the device. **Operational** Send command to the specific field device to go into operational mode. **Pre-Operational** Send command to the specific field device to go into pre-operational mode. **Stop** Send command to the specific field device to go into stop mode. **Reset** Send command to the specific field device to reset.

	Send Store Parameters Send the write command to parameter index 1010 sub-index 2, to save all parameters in the CANopen Slave device to non-volatile memory.

Table 5.8. - CANopen Slave Device Online Status

5.4. 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.

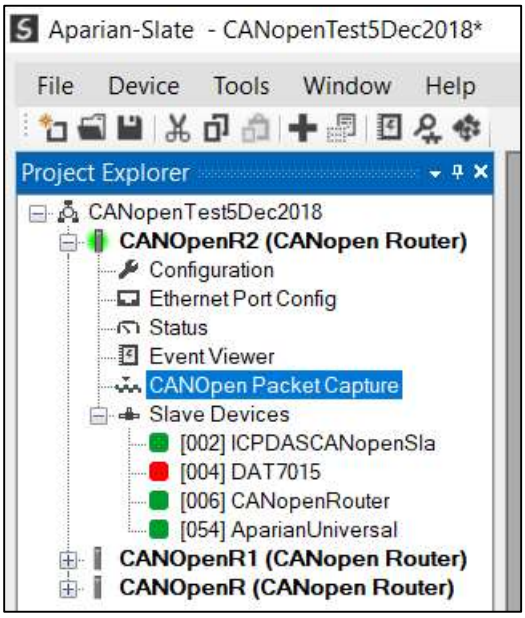


Figure 5.11 - Selecting CANopen Packet Capture

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

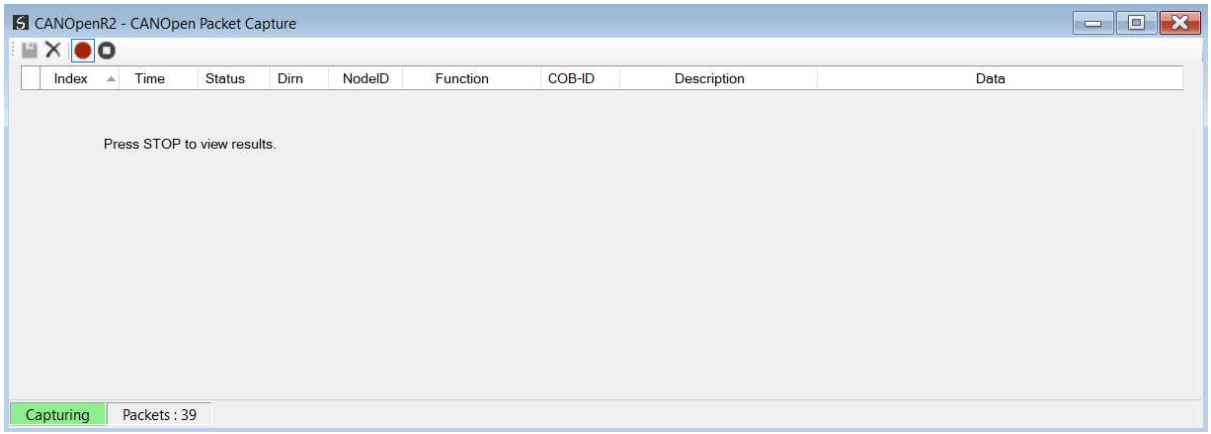


Figure 5.12 – CANopen packet capture

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

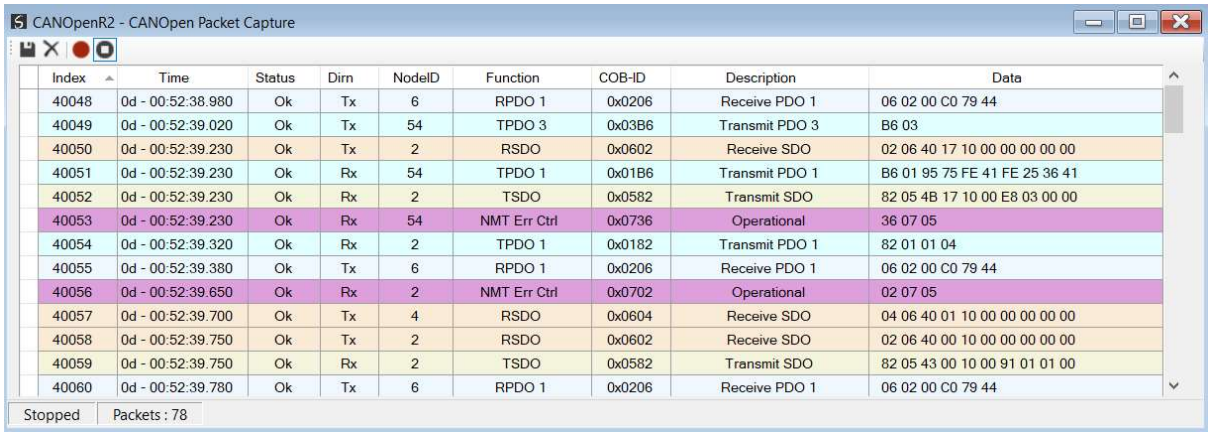


Figure 5.13 – CANopen Packet Capture complete

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 5.9 – 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.

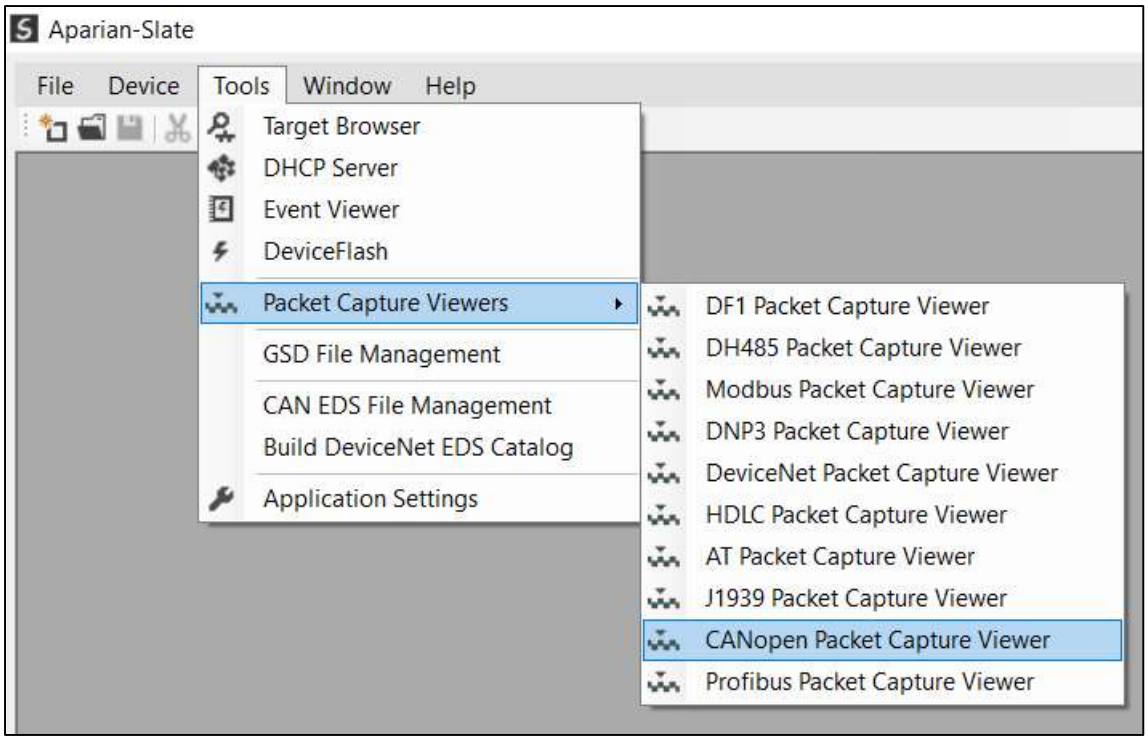


Figure 5.14 - Selecting the CANopen Packet Capture Viewer

5.5. MODULE EVENT LOG

The CANopen Router 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.

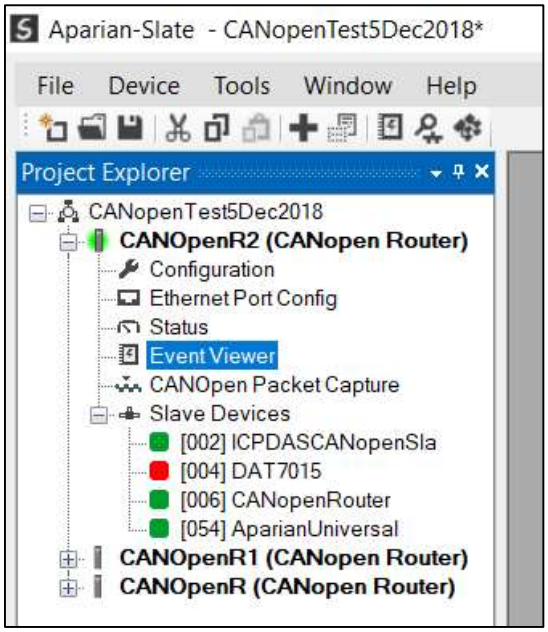
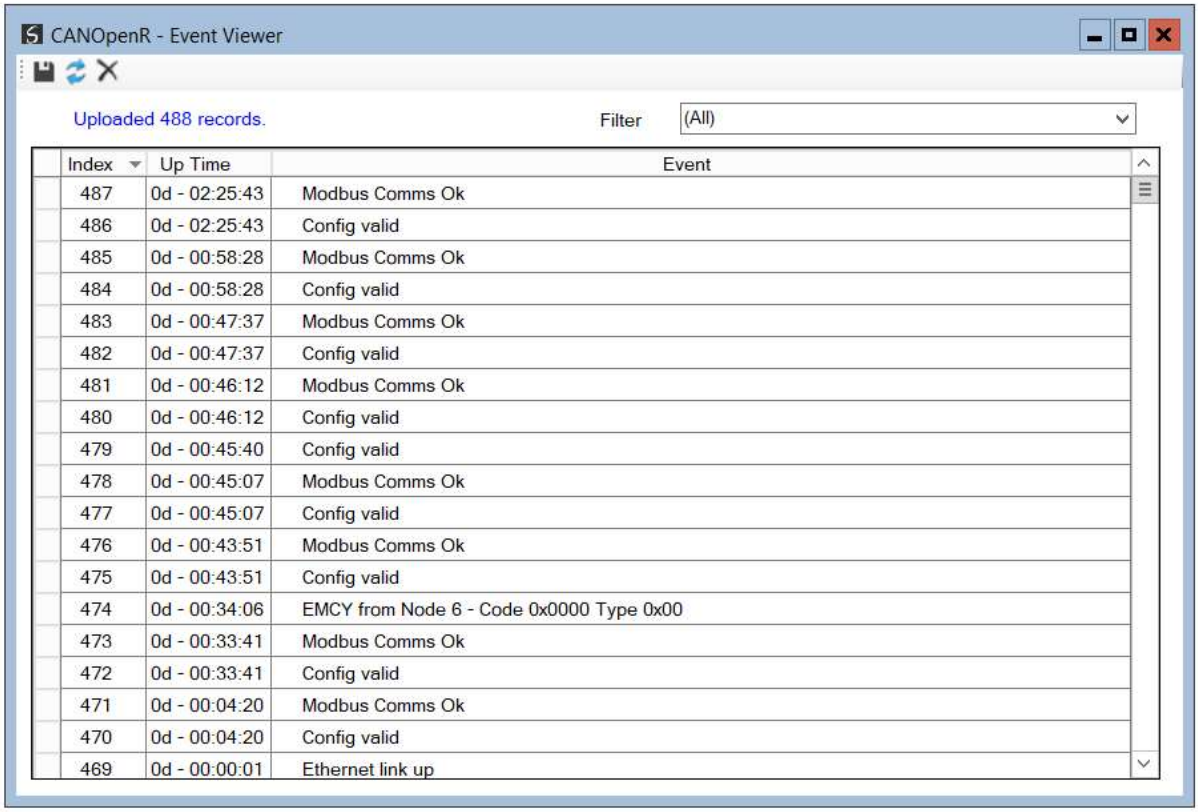


Figure 5.15. - Selecting the module Event Log

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
487	0d - 02:25:43	Modbus Comms Ok
486	0d - 02:25:43	Config valid
485	0d - 00:58:28	Modbus Comms Ok
484	0d - 00:58:28	Config valid
483	0d - 00:47:37	Modbus Comms Ok
482	0d - 00:47:37	Config valid
481	0d - 00:46:12	Modbus Comms Ok
480	0d - 00:46:12	Config valid
479	0d - 00:45:40	Config valid
478	0d - 00:45:07	Modbus Comms Ok
477	0d - 00:45:07	Config valid
476	0d - 00:43:51	Modbus Comms Ok
475	0d - 00:43:51	Config valid
474	0d - 00:34:06	EMCY from Node 6 - Code 0x0000 Type 0x00
473	0d - 00:33:41	Modbus Comms Ok
472	0d - 00:33:41	Config valid
471	0d - 00:04:20	Modbus Comms Ok
470	0d - 00:04:20	Config valid
469	0d - 00:00:01	Ethernet link up

Figure 5.16. – 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.

5.6. WEB SERVER

The CANopen Router 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.

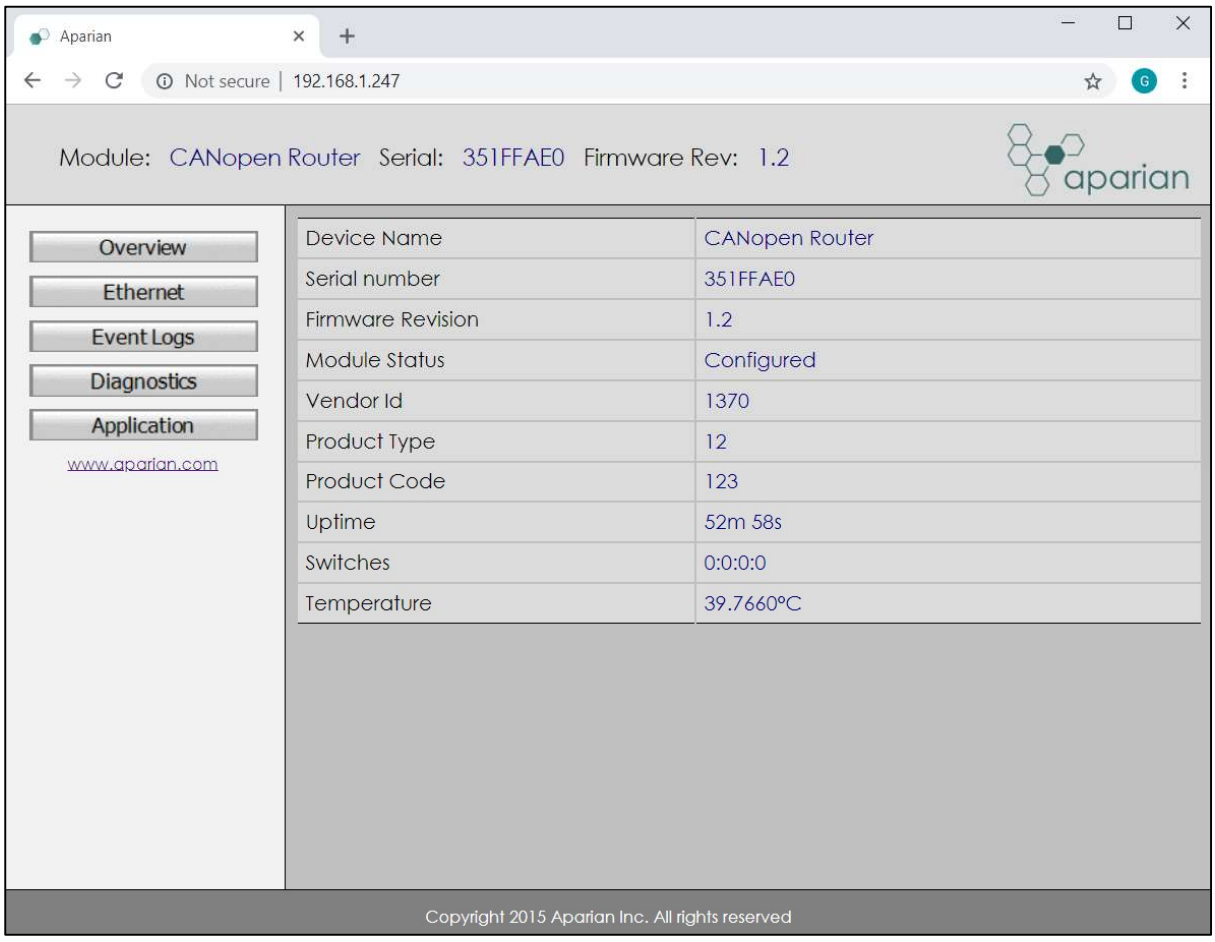


Figure 5.17. - Web interface

6. TECHNICAL SPECIFICATIONS

6.1. DIMENSIONS

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

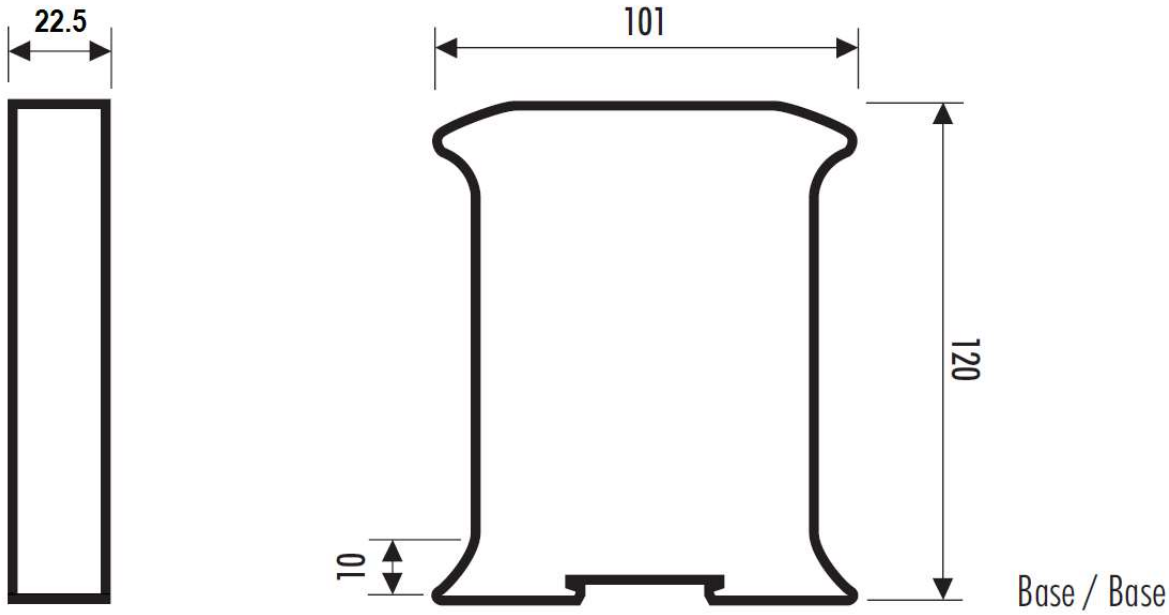


Figure 6.1 – CANopen Router enclosure dimensions

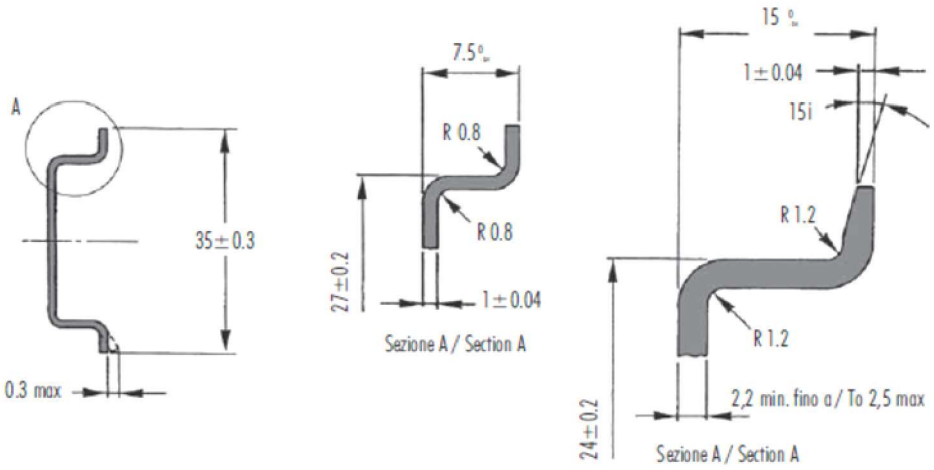


Figure 6.2 - Required DIN dimensions

6.2. ELECTRICAL

Specification	Rating
Power requirements	Input: 10 – 28V DC, (70 mA @ 24 VDC)
Power consumption	1.7 W
Connector	5-way terminal, 5.08mm pitch.
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 6.1 - Electrical specification

6.3. ETHERNET

Specification	Rating
Connector	RJ45
Conductors	CAT5 STP/UTP
ARP connections	Max 20
TCP connections	Max 20
CIP connections	Max 10
Communication rate	10/100Mbps
Duplex mode	Full/Half
Auto-MDIX support	Yes

Table 6.2 - Ethernet specification

6.4. CANOPEN NETWORK

Specification	Rating
Connector	5-way terminal, 5.08mm pitch.
Modes	CANopen Master CANopen Slave
CANopen Slave Count	64
PDO Count per Device	16
Supported Baud Rates	50k 125k 250k 500k 800k 1M
CiA 443 Support	Yes
NMT messages	Operational Control (e.g. Stopped, Pre-operational, Operational) SYNC TIME EMCY

Table 6.3 – CANopen specification

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

6.5. 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

Table 6.4 – Certifications

7. INDEX