CAN Bus Router

User Manual

A-CANBR/B

Document No. D165-007
04/2026
Revision 1.4

aparian

CONTENTS

1. Preface 7
1.1. Introduction to the CAN Bus Router 7
1.2. Features 9
1.3. Architecture 10
1.4. Additional Information 11
1.5. Support 11
2. Installation 13
2.1. Module Layout 13
2.1. Module Mounting 15
2.2. Bottom Power 16
2.3. RS232/RS485 Port 17
2.4. RS485 Termination 17
2.5. Ethernet Ports 18
2.6. CAN and Front Power 18
2.7. North American Hazardous Location Approval 18
3. Setup 20
3.1. Install Configuration Software 20
3.2. Network Parameters 20
3.3. Creating a New Project 25
3.4. General parameters 27
3.5. CAN Bus Configuration 29
3.5.1. CAN Receive 30
3.5.2. CAN Send 32
3.6. Primary Interface 34
3.6.1. EtherNet/IP Target 34
3.6.1.1. Studio / Logix 5000 Configuration 35
3.6.1.2. Internal Data Space Mapping 44
3.6.2. Modbus Server and Client 45
3.6.2.1. Modbus Server 45
3.6.2.2. Modbus Client 45
3.6.2.3. Modbus Configuration 45

3.6.2.4. Internal Data Space Mapping 47
3.6.2.5. Modbus Auxiliary Map 49
3.6.3. EtherNet/IP Originator 50
3.6.3.1. EtherNet/IP Class 1 Device Connections 51
3.6.3.2. EtherNet/IP Explicit Message Device Connections 60
3.6.3.3. Internal Data Space Mapping 64
3.7. Internal Data Space Map 66
3.7.1. Copy From 67
3.7.1.1. Internal 67
3.7.1.2. EIP Target 67
3.7.1.3. EIP Originator 68
3.7.1.4. Modbus Register 69
3.7.1.5. System 69
3.7.2. Copy To 71
3.7.2.1. Internal 71
3.7.2.2. EIP Target 72
3.7.2.3. EIP Originator 72
3.7.2.4. Modbus Register 73
3.8. Constants 73
3.9. Advanced 74
3.10. Module Download 75
4. SD Card 78
4.1. Firmware 78
4.2. Network Parameters 79
4.3. Configuration 80
4.3.1. Manual Copy 81
4.3.2. Slate Triggered Upload 82
5. Device Firmware Update 84
6. Operation 87
6.1. CAN Bus 87
6.1.1. CAN Receive 87
6.1.2. CAN Send 88
6.1.3. Explicit CAN Messaging 89

6.2. EtherNet/IP Target 91
6.2.1. Class 1 Assembly Mapping 91
6.2.2. Explicit Messaging 93
6.2.2.1. Explicit CAN Message 93
6.2.2.2. Set CAN Baud Rate 95
6.3. EtherNet/IP Originator 96
6.3.1. EtherNet/IP Class 1 Connections 96
6.3.2. Explicit EtherNet/IP Messaging 97
6.4. Modbus Client 98
6.5. Modbus Server 100
7. Diagnostics 102
7.1. LEDs 102
7.2. Module Status Monitoring in Slate 103
7.2.1. General 104
7.2.2. CAN Statistics 108
7.2.3. CAN Receive Status 109
7.2.4. CAN Send Status 110
7.2.5. EtherNet/IP Explicit 110
7.2.6. EtherNet/IP Map 112
7.2.7. EtherNet/IP Originator 112
7.2.8. Logix 113
7.2.9. Modbus 114
7.2.10. Serial Statistics 116
7.2.11. CIP Statistics 117
7.2.12. Ethernet Clients 118
7.2.13. TCP/ARP 118
7.3. Target Device Status Monitoring In Slate 119
7.3.1. EtherNet/IP 119
7.3.1.1. General 119
7.3.1.2. Input Data 121
7.3.1.3. Output Data 121
7.4. Module Event Log 122
7.5. Web Server 123

7.6. CAN Bus Packet Capture 123
7.7. Modbus Packet Capture 126
7.8. Module Status Report 129
7.9. Modbus Summary CSV 130
7.10. Modbus Expanded CSV 131
7.11. Internal Register Viewer 132
8. Technical Specifications 134
8.1. Dimensions 134
8.2. Electrical 135
8.3. Ethernet 135
8.4. Serial Port (RS232) 136
8.5. Serial Port (RS485) 136
8.6. CAN Bus 137
8.7. EtherNet/IP Target 137
8.8. EtherNet/IP Originator 137
8.9. Modbus Client 138
8.10. Modbus Server 138
8.11. Certifications 139
9. Index 140

Revision History

Revision	Date	Comment
1.0	29 August 2024	Initial document
1.1	6 May 2025	Fixed section numbers in the Diagnostics
1.2	20 May 2025	Add duplicate IP address indication to LEDs in the Diagnostics section.
1.3	28 January 2026	Added section on the Internal Register Viewer. Add new RS232 and RS485 statistics.
1.4	20 April 2026	Added UL North American Hazardous Location Approval.

1. PREFACE

1.1. INTRODUCTION TO THE CAN BUS ROUTER

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

Primary Interface:
The module supports one of four Primary Interface modes:

EtherNet/IP Target

Here a remote EtherNet/IP device (e.g. a Logix controller) establishes a number of Class 1 connections to the module. CAN Bus data can be mapped between the input and output class 1 cyclic connections of the Logix controller (allowing up to 2KB input and 2KB output to be exchanged at the requested packet interval – RPI).

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

- **EtherNet/IP Explicit Messaging**

  This allows CAN Bus devices to exchange data with up to 10 EtherNet/IP devices. The module can use either Class 3 or Unconnected Messaging (UCMM) to Get and Set data in the remote EtherNet/IP devices.

- **Direct-To-Tag technology**

  This allows the CAN Bus devices to exchange data with a Logix controller without the need to write any application code (e.g. ladder) in Studio 5000. The CAN Bus data is directly read from, or written to, Logix tags.

- **EtherNet/IP Class 1 connection**

  CAN Bus data can be mapped to a maximum of 10 EtherNet/IP devices using input and output class 1 cyclic connections. This will allow the CAN Bus Router to “own” the EtherNet/IP target device and exchange CAN Bus data using the EtherNet/IP device’s input and output assemblies.

Modbus Server
The diagnostics and CAN Bus data will be written to, or read from, the module’s internal Modbus Registers (Holding or Input Registers). These registers can be accessed by a remote Modbus Client using either Modbus TCP, Modbus RTU232, or Modbus RTU485.

Modbus Client

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

When configured as an EtherNet/IP Target, the software can generate an L5X file which can be imported into Studio 5000. This provides all the necessary UDTs, Tags and mapping ladder to allow the Logix application code to easily access the CAN Bus data.

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

The module also provides a range of statistics, CAN Bus and Modbus packet capture functions, and internal Modbus and Data table reads to simplify remote diagnosis.

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

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

1.2. FEATURES

- Receive up to 100 pre-configured CAN Bus packets.
- Send up to 100 pre-configured CAN Bus packets.
- CAN Bus transmission can be scheduled (periodic), or externally triggered, sent on Start-up, or any combination of these.
- Logix L5X generation - Ready for import
- Supports Passthrough Messaging from Logix
- Module has various Primary Interfaces:
    - EtherNet/IP Target (Class 1 connection)
    - Modbus Server (TCP, RTU232, and RTU485)
    - Modbus Client (TCP, RTU232, and RTU485)
    - EtherNet/IP Originator (Class 1 connection with up to 10 EtherNet/IP devices and Explicit Messaging, including Direct-To-Tag Logix tag access, with up to 10 EtherNet/IP devices).
- Slate software provides a CAN Bus and Modbus packet capture utility for better diagnosis of communication issues.
- Supports common CAN Bus Baud Rates (10k, 20k, 50k, 125k, 250k, 500k, 800k, 1M)
- Dual Ethernet ports which support Device-Level-Ring (DLR).
- Network Time Protocol (NTP) supported for external time synchronization.
- Small form factor – DIN rail mounted.

1.3. ARCHITECTURE

The figures below provide an example of the typical network setup for connecting CAN Bus devices to either EtherNet/IP or Modbus TCP/RTU232/RTU485.

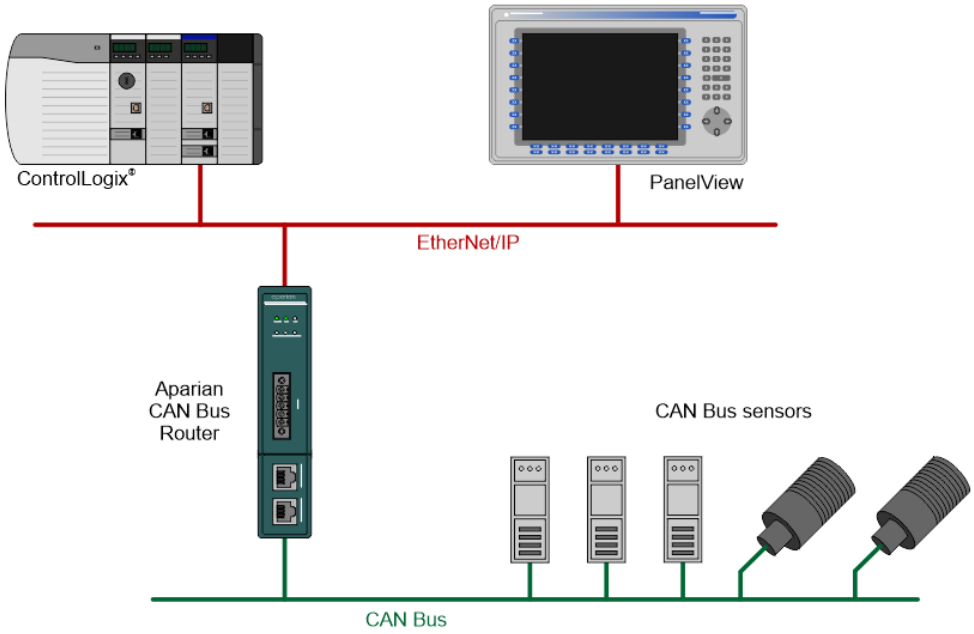


Figure 1.1. – Example of connecting CAN Bus sensors to a Logix Controller

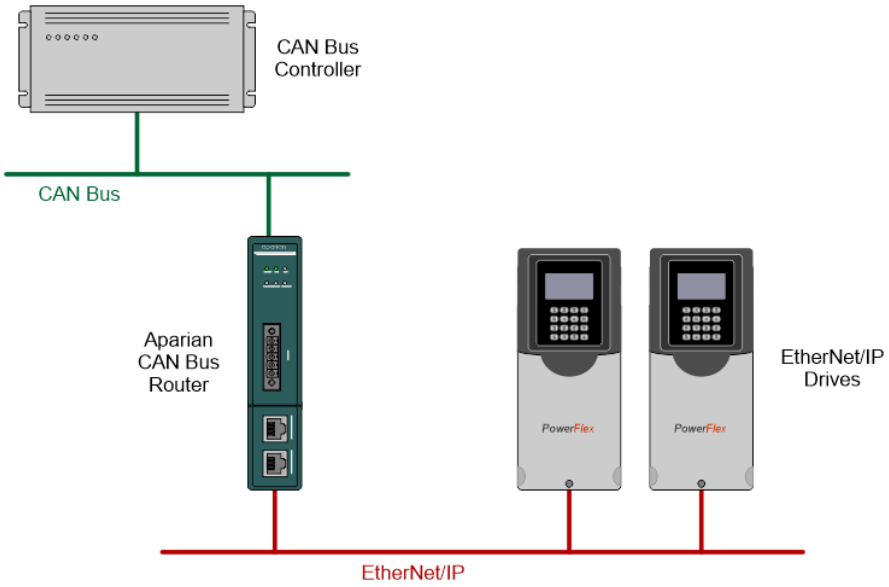


Figure 1.2. – Example of connecting EtherNet/IP drives to a CAN Bus network

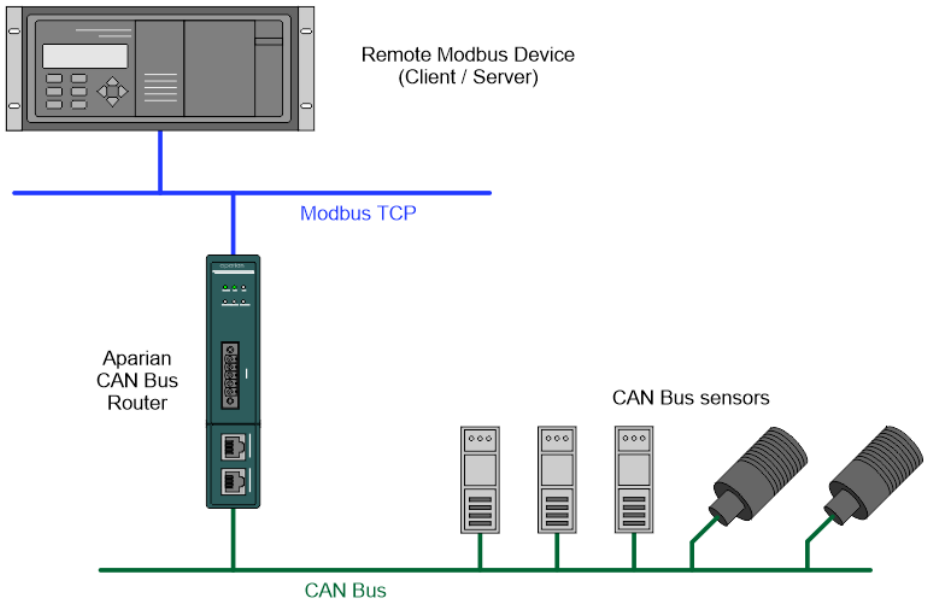


Figure 1.3. - Example of connecting CAN Bus devices to a Modbus TCP Client or Server

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
CAN Bus Router User Manual CAN Bus Router Datasheet Example Code & UDTs	https://www.aparian.com/products/CANbusRouterb
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

Table 1.1. - Additional Information

1.5. SUPPORT

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

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

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

Table 1.2. – Support Details

2. INSTALLATION

2.1. MODULE LAYOUT

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

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

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

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

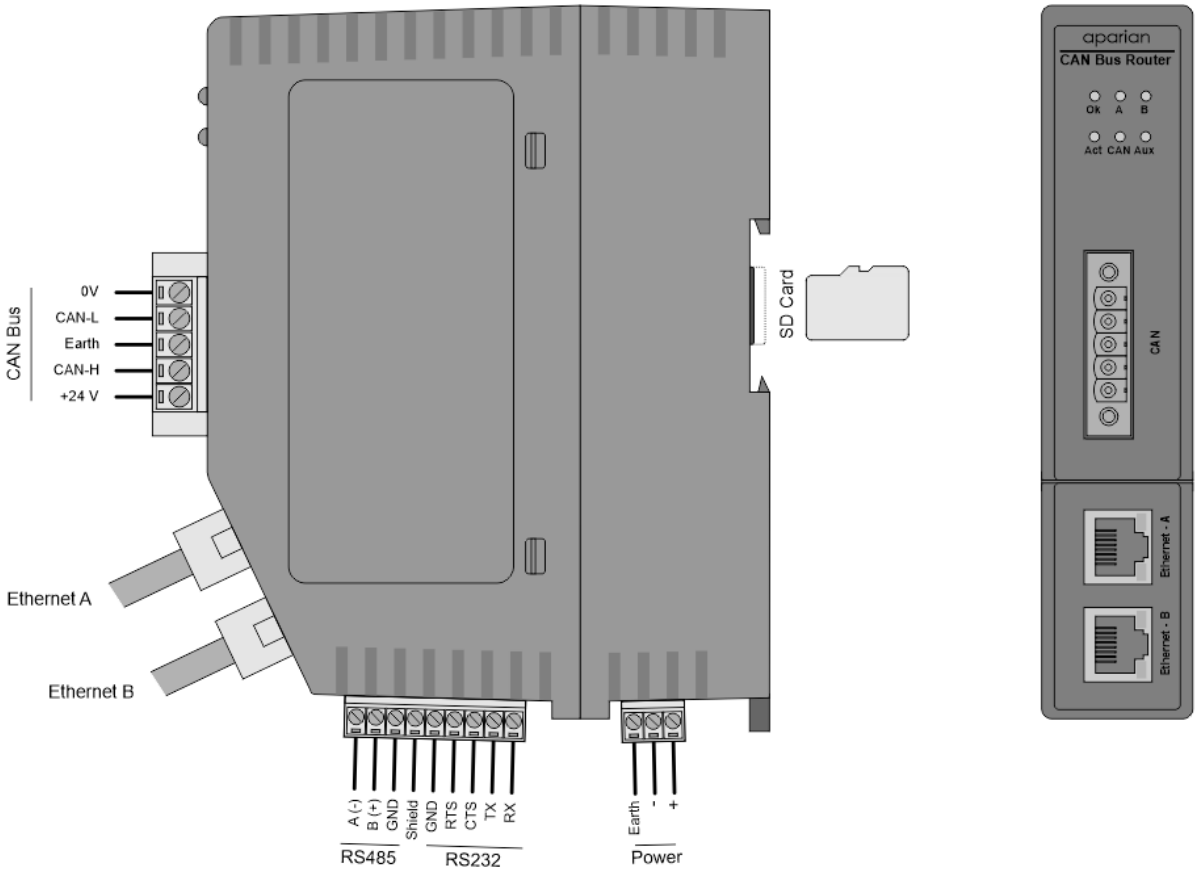


Figure 2.1 – CAN Bus Router side and front view

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

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

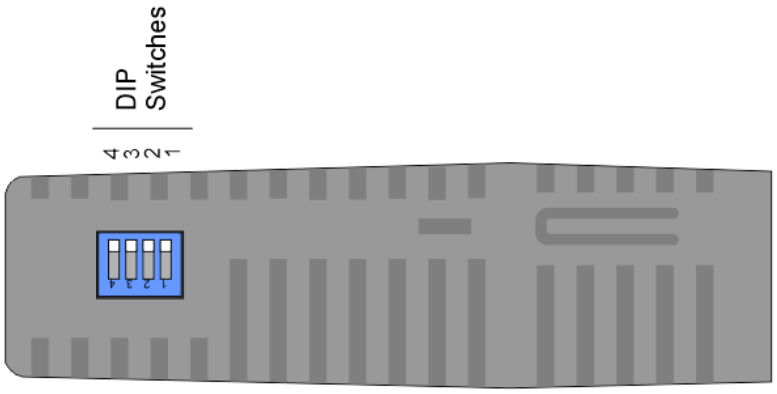


Figure 2.2 – CAN Bus Router top view

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

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

Table 2.1 - DIP Switch Settings

2.1. MODULE MOUNTING

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

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

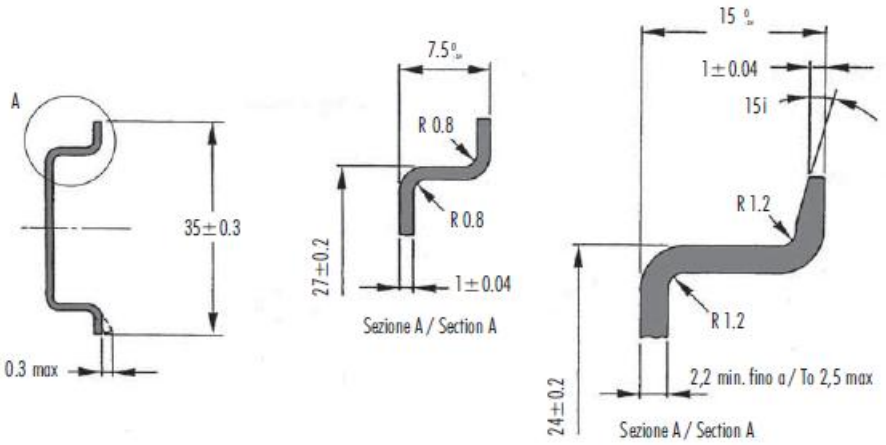


Figure 2.3 - DIN rail specification

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

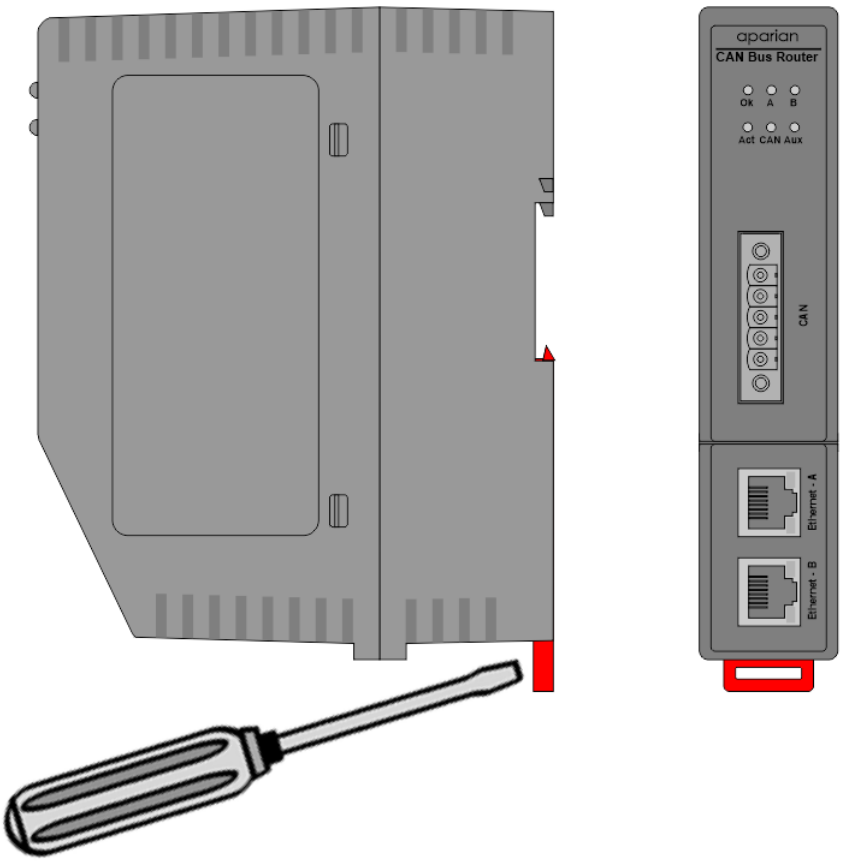


Figure 2.4 - DIN rail mouting

2.2. BOTTOM POWER

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

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

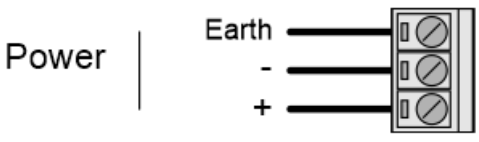


Figure 2.5 - Power connector

2.3. RS232/RS485 PORT

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

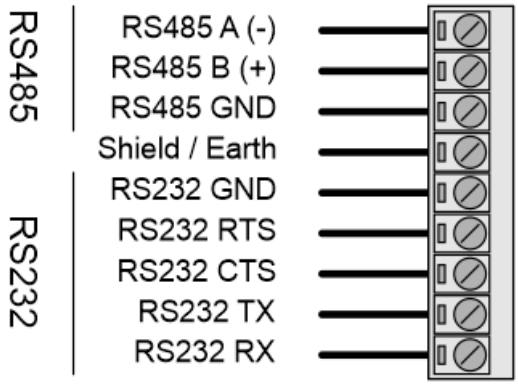


Figure 2.6 - RS232/RS485 connector

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

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

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

2.4. RS485 TERMINATION

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

2.5. ETHERNET PORTS

The Ethernet connectors should be wired according to industry standards. **Refer** to the additional information section in this document for further details. The module has an embedded switch connecting the two Ethernet ports.

2.6. CAN AND FRONT POWER

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

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

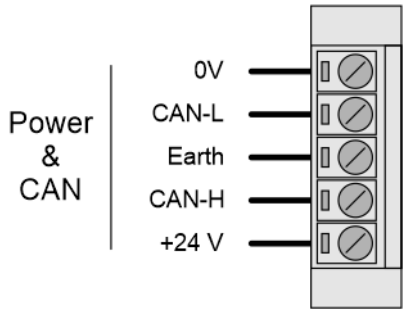


Figure 2.7 – CAN and Power connector

2.7. NORTH AMERICAN HAZARDOUS LOCATION APPROVAL

The following information applies when operating the product in hazardous locations:

SUITABLE FOR USE IN CLASS I, DIVISION 2, GROUPS A, B, C AND D HAZARDOUS LOCATIONS, OR NONHAZARDOUS LOCATIONS ONLY.

-20°C to +70°C, T5A
Class 1, Div 2 Gps A, B, C, D

THIS EQUIPMENT IS OPEN-TYPE DEVICE AND IS MEANT TO BE INSTALLED IN AN ENCLOSURE SUITABLE FOR THE ENVIRONMENT SUCH THAT THE EQUIPMENT IS ONLY ACCESSIBLE WITH THE USE OF A TOOL.

WARNING - EXPLOSION HAZARD - DO NOT DISCONNECT EQUIPMENT WHILE THE CIRCUIT IS LIVE OR UNLESS THE AREA IS KNOW TO BE FREE OF IGNITABLE CONCENTRATIONS.

WARNING - EXPLOSION HAZARD - SUBSTITUTION OF ANY COMPONENT MAY IMPAIR SUITABILITY FOR CLASS I, DIVISION 2.

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.

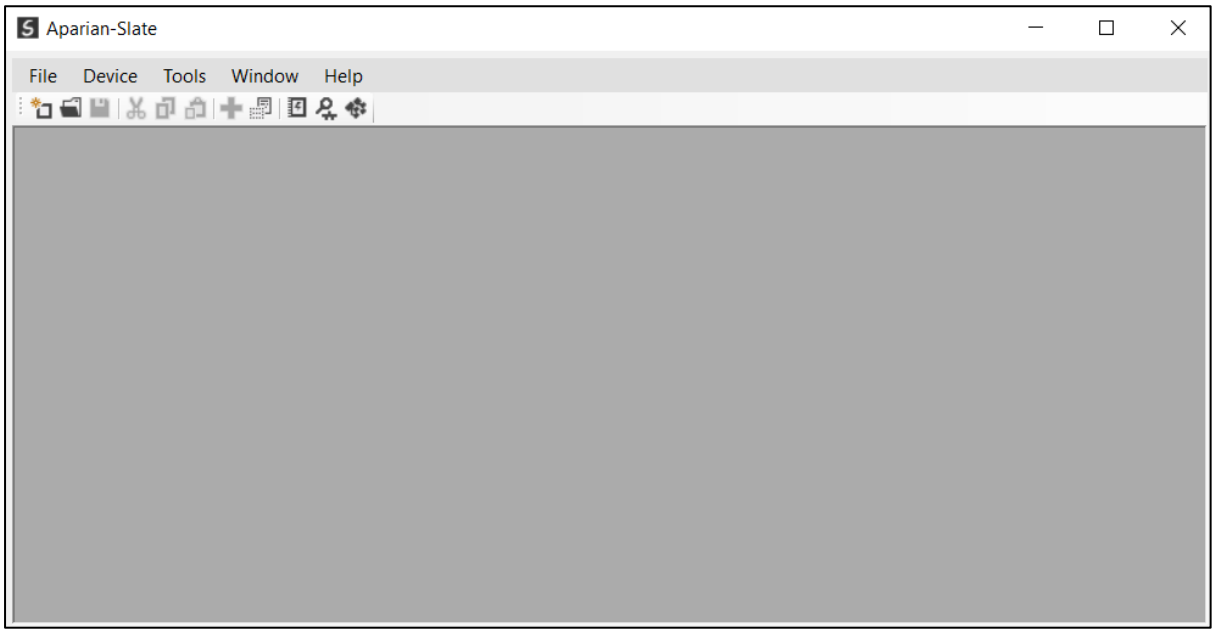


Figure 3.1. - Aparian Slate Environment

3.2. NETWORK PARAMETERS

The module will have DHCP (Dynamic Host Configuration Protocol) enabled as factory default. Therefore, 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.

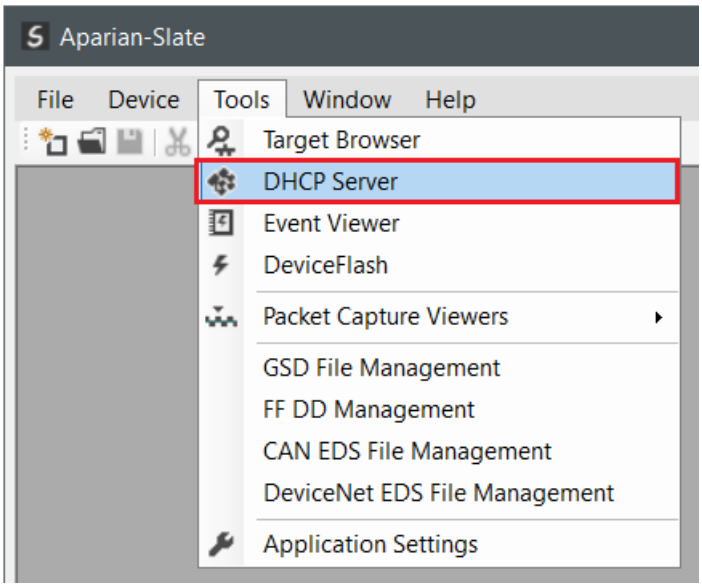


Figure 3.2. - Selecting DHCP Server

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

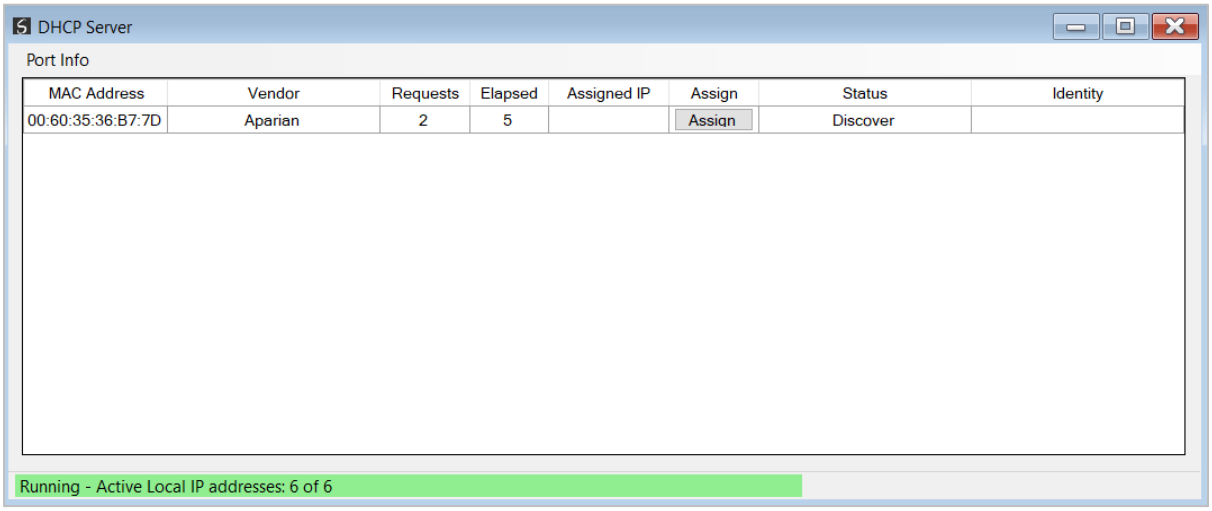


Figure 3.3. - DHCP Server

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.

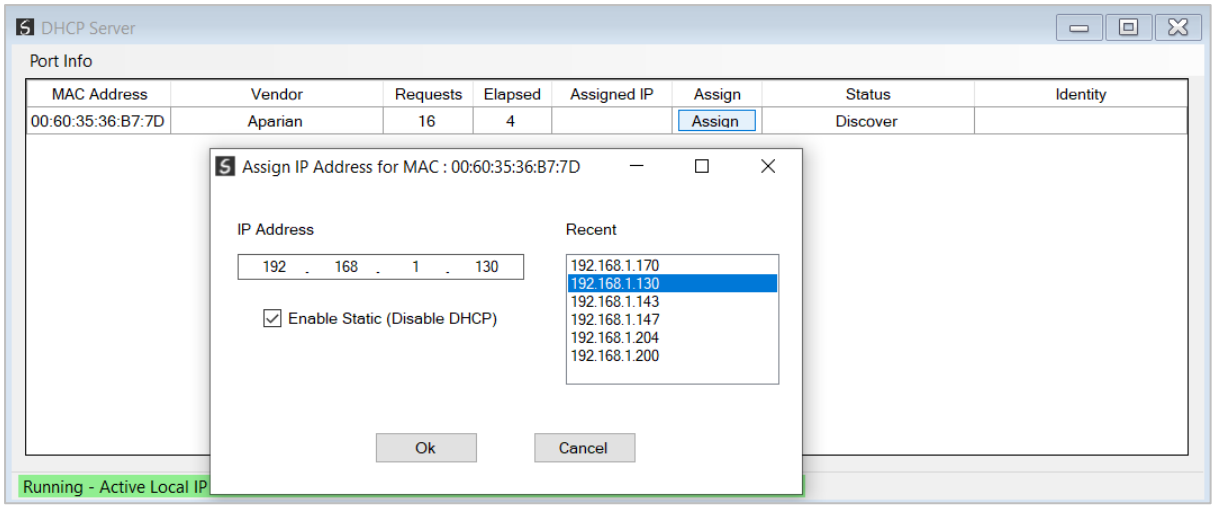


Figure 3.4. - Assigning IP Address

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's 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.

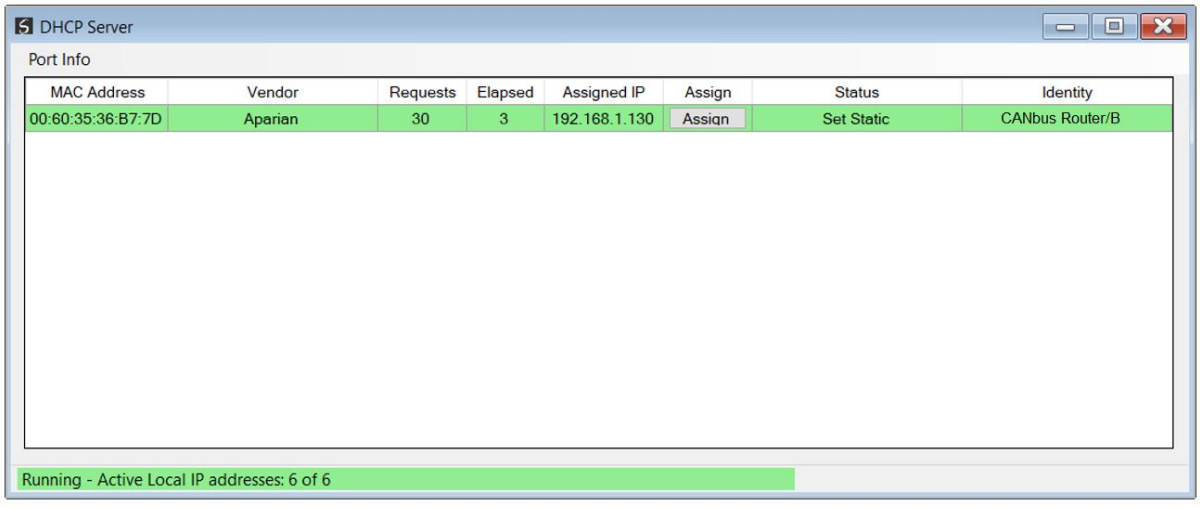


Figure 3.5. - Successful IP address assignment

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.

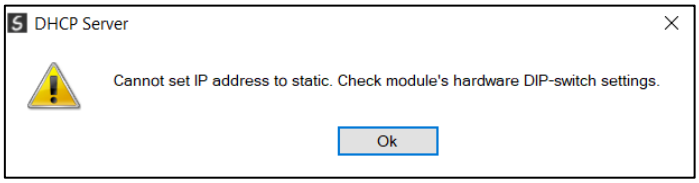


Figure 3.6. - Force DHCP warning

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.

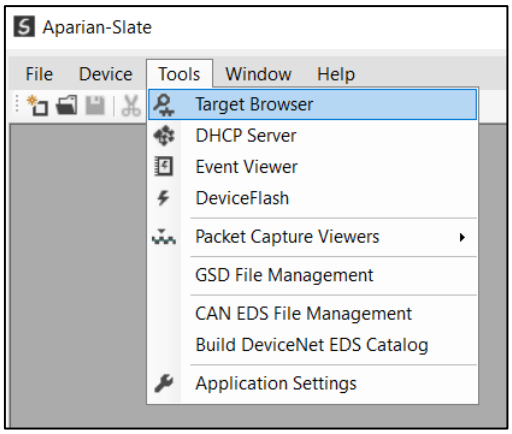


Figure 3.7. - Selecting the Target Browser

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

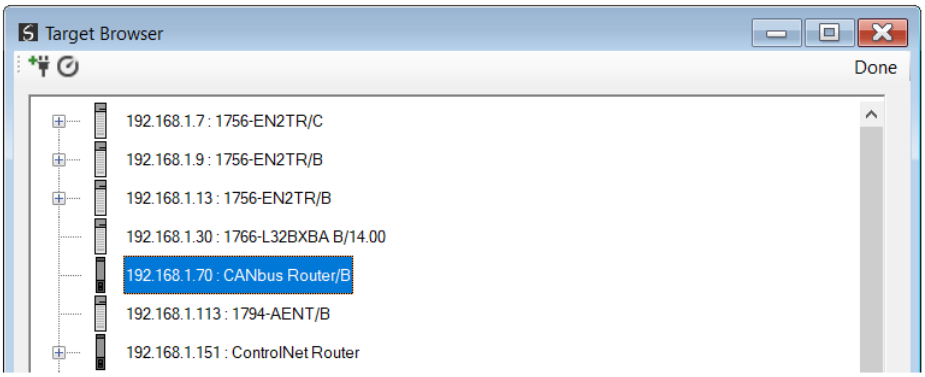


Figure 3.8. - Target Browser

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

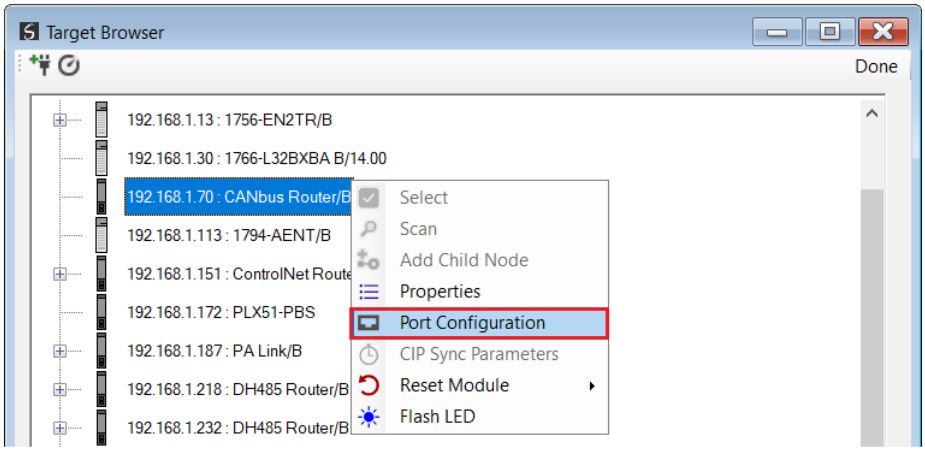


Figure 3.9. - Selecting Port Configuration

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

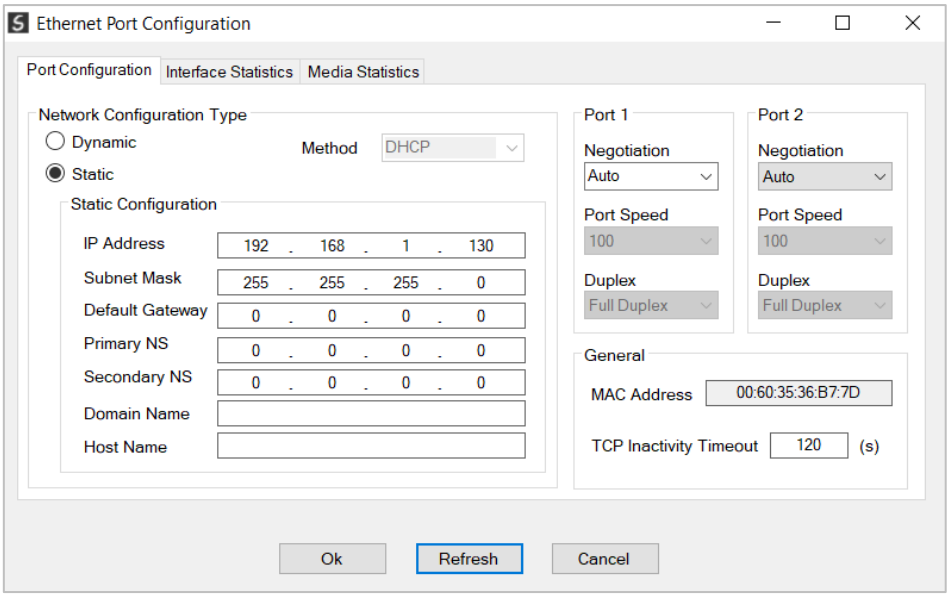


Figure 3.10. - Port Configuration

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***.

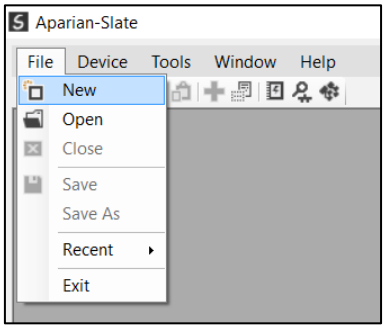


Figure 3.11. - Creating a new project

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.

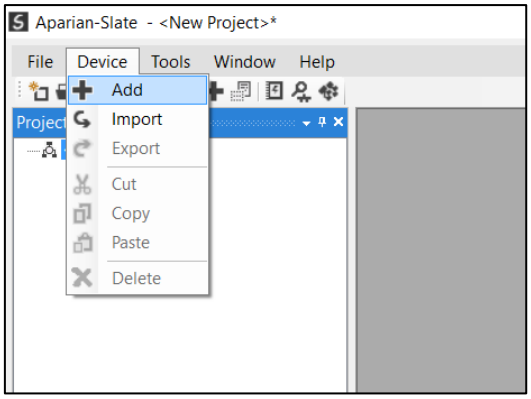


Figure 3.12. - Adding a new device

In the ***Add New Device*** window select the ***CAN Bus Router*** and click the ***Ok*** button.

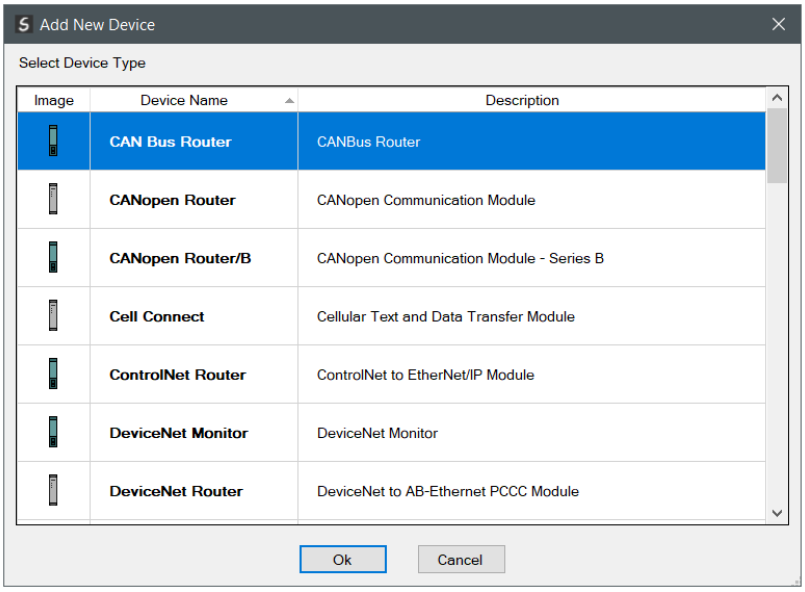


Figure 3.13 – Selecting a new CAN Bus Router

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***.

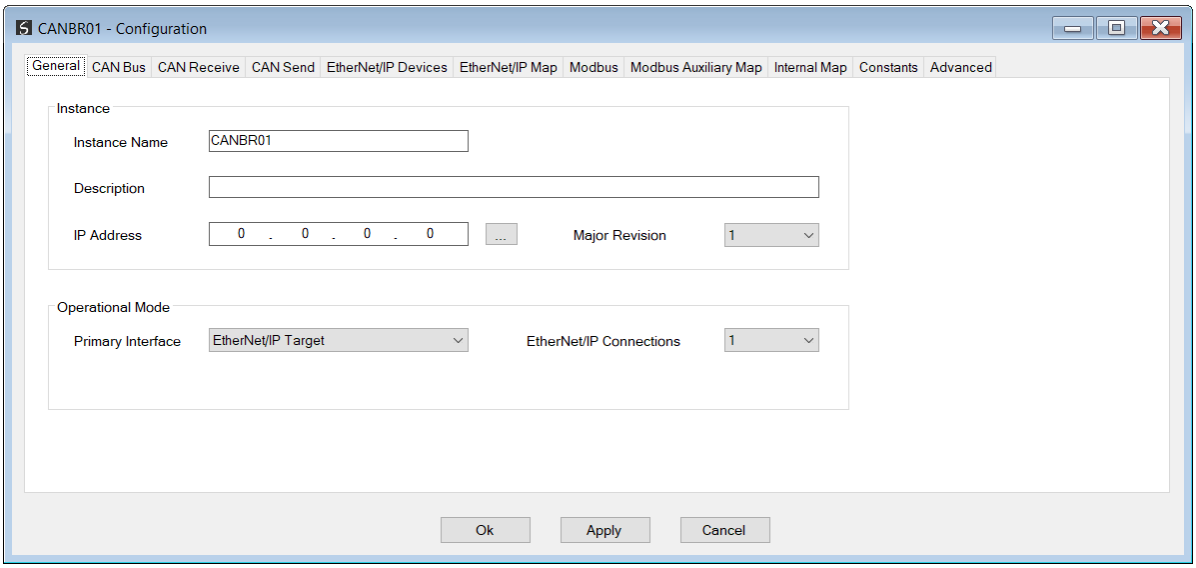


Figure 3.14. – CAN Bus Router configuration

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

3.4. GENERAL PARAMETERS

The CAN Bus parameters will be configured by Slate. 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 the module configuration stored in non-volatile storage.

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

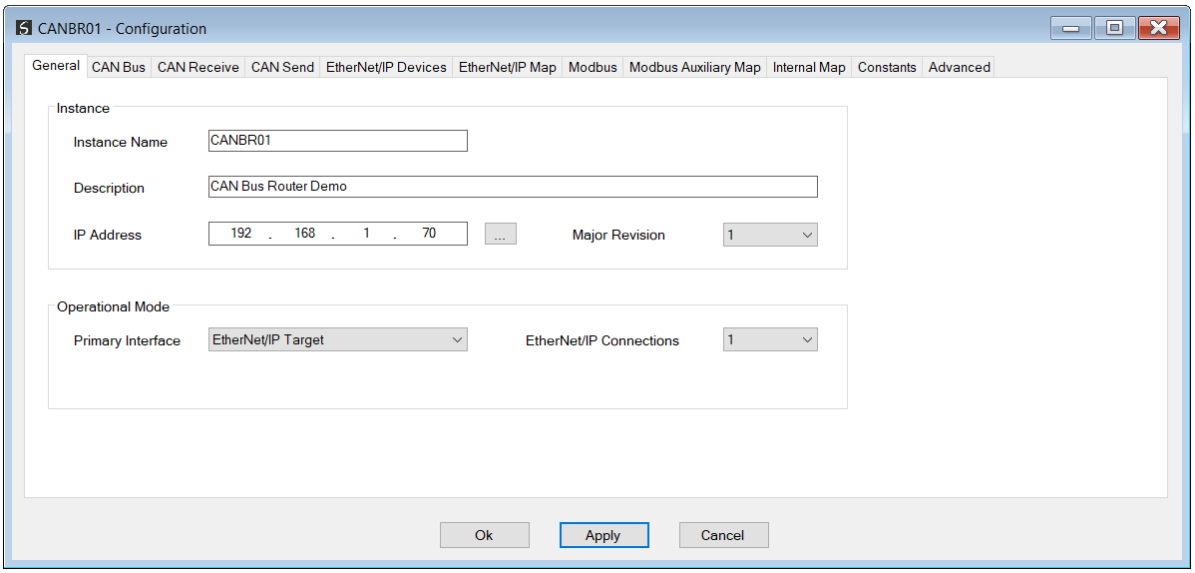


Figure 3.15 - General Configuration

The general configuration consists of the following parameters:

Parameter	Description
Instance Name	This parameter is a user defined name to identify between various CAN Bus Routers.
Description	This parameter is used to provide a more detailed description of the application for the module.
Major Revision	The major revision of the module
EtherNet/IP Connections	The number of connections the module will use when operating as an EtherNet/IP target.
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 CAN Bus Router on the network.
Primary Interface	**EtherNet/IP Target** A Logix controller can own the CAN Bus Router over EtherNet/IP using up to 4 class 1 connections. **Modbus Server** A Modbus Client can read and write data to the module which can then be mapped to one or more CAN Bus devices. The module can operate as a Modbus Server on Ethernet TCP, RTU232, and RTU485 **Modbus Client** A module can read and write data from various Modbus devices which can then be mapped to one or more CAN Bus devices. The module can operate as a Modbus Client on Ethernet TCP, RTU232, and RTU485 **EtherNet/IP Originator** As an EtherNet/IP originator, the module can use two methods to read and write data to and from an EtherNet/IP device (IO):

	EtherNet/IP Class 1 Connection The CAN Bus Router can own EtherNet/IP IO by using the Slate software to configure the IO connections. *EtherNet/IP Explicit Messaging* The CAN Bus Router can exchange data with up to 10 EtherNet/IP devices using explicit messaging.

Table 3.1 - General configuration parameters

3.5. CAN BUS CONFIGURATION

The ***CAN Bus*** communication configuration window is opened by either double clicking on the module in the tree, or right-clicking the module and selecting ***Configuration*** and then selecting the ***CAN Bus*** tab.

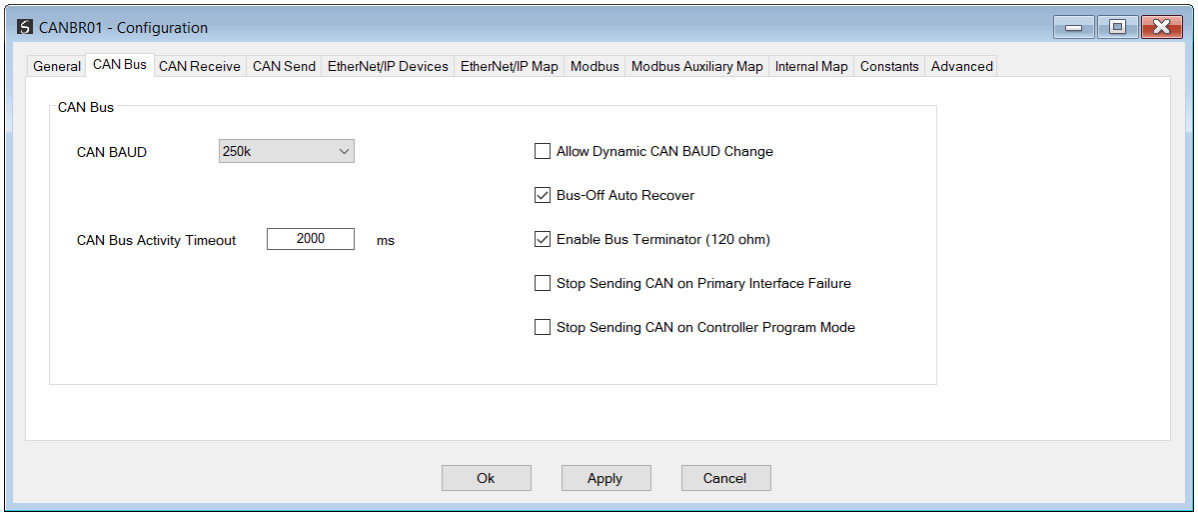


Figure 3.16 – CAN Bus communication Configuration

The **CAN Bus Communication** configuration consists of the following parameters:

Parameter	Description
BAUD Rate	The CAN Bus BAUD rate. The following options are available: • 10k • 20k • 50k • 125k • 250k • 500k • 800k • 1M

CAN Bus Activity Timeout	The amount of time (in milliseconds) without receiving any valid configured CAN Receive packets before the originating Primary Interface (EtherNet/IP Originator or Modbus Client) is suspended.
Allow Dynamic CAN BAUD Change	When enabled, the module will support the dynamic change of the CAN Bus Baud Rate using a specific CIP message. See the Operation section for more details regarding the message. **NOTE:** When the CAN Bus Router module is power cycled, the CAN Baud Rate will return to the Baud Rate configured in the Slate configuration and the Baud Rate set dynamically will **not** persist.
Bus-Off Auto Recover	Enables automatic CAN Bus network recovery when the module has detected that the CAN Bus network is off.
Enable Bus Terminator	Enables the internal 120 Ohm terminator on the CAN Bus network. The CAN Bus network must be terminated at the two extremities of the network (i.e., the start and end of the network).
Stop Sending CAN on Primary Interface Failure	When this option is selected, the CAN Bus Router will stop sending configured CAN packets when the primary interface communication has failed.
Stop Sending CAN on Controller Program Mode	When this option is selected and the primary interface has been set to EtherNet/IP Target, the CAN Bus Router will stop sending configured CAN packets when the controller (which is the EtherNet/IP communication originator) is in Program mode.

Table 3.2 - CAN Bus Communication configuration parameters

3.5.1. CAN RECEIVE

The CAN Receive configuration is used to determine which received CAN Bus packets are processed by the CAN Bus Router.

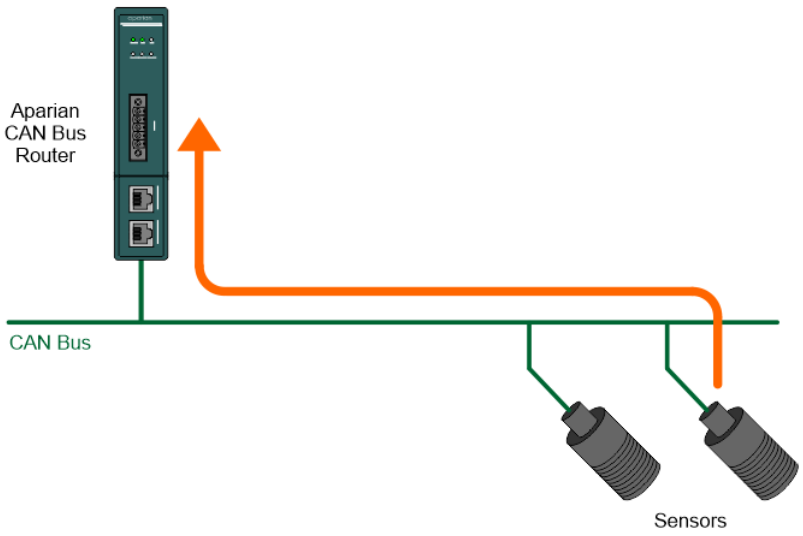


Figure 3.20 – CAN Receive data flow

The ***CAN Receive*** configuration window is opened by either double clicking on the module in the tree, or right-clicking the module and selecting ***Configuration*** and then selecting the ***CAN Receive*** tab.

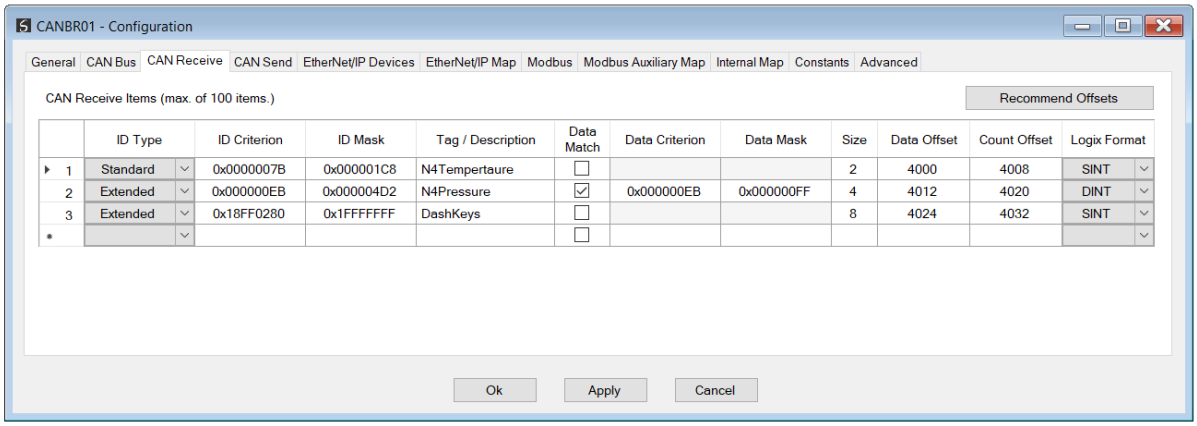


Figure 3.17 – CAN Receive Configuration

The **CAN Receive** configuration consists of the following parameters:

Parameter	Description
ID Type	The CAN ID Type, either: • Standard – (11-bit CAN Header) • Extended – (29-bit CAN Header)
ID Criterion	The CAN ID required in a received message to be matched with this transaction. The received CAN ID, and ID Criterion are both masked (bitwise-AND) with the ***ID Mask***, before making the match comparison. This allows a range of CAN IDs to be accepted as a valid response.
ID Mask	The CAN ID Mask to be used (bitwise-AND) when matching a received message. Each bit in the mask corresponds to the same bit location in the ***ID Criterion*** and received CAN ID, where: Mask Bit = 0: the bit in received message CAN-ID is ignored. Mask Bit = 1: the bit in the received message CAN-ID must equal the corresponding bit in the ***ID Criterion***. For example, if: Configured ***ID Criterion*** = 0x354 Configured ***ID Mask*** = 0xFFC Then, any message with a CAN ID of either: 0x354, 0x355, 0x356 or 0x357 will be a match.
Tag / Description	A user defined Tag or Description string for the CAN ID. The maximum length of the string is 32 characters.

	Note: When configured as an EtherNet/IP Target, this field will be used in the generation of the Logix tags and must therefore comply with the Logix tag naming rules. (No extended characters or control characters etc.)
Data Match	Enables CAN Data criteria. When enabled, not only must the CAN ID match, but also the payload data. The received CAN data payload, and configured ***Data Criterion*** are both masked (bitwise-AND) with the configured ***Data Mask***, before making the match comparison. This allows a range of data payloads to be accepted.
Data Criterion	The required CAN data payload (masked with ***Data Mask***) for a match.
Data Mask	The CAN payload data mask to be used (bitwise-AND) when matching a received message. Each bit in the mask corresponds to the same bit location in the ***Data Criterion*** and received CAN payload data, where: Mask Bit = 0: the bit in received message CAN payload data is ignored. Mask Bit = 1: the bit in the received message CAN payload data must equal the corresponding bit in the ***Data Criterion***.
Size	The required size (in bytes) of the receive CAN packet payload. (0-8)
Data Offset	The offset in the Internal Data Space where the received CAN data will be copied after a successful (CAN ID and Data) match.
Count Offset	The offset in the Internal Data Space where the received CAN item transaction count will be copied. (4 bytes). This count is incremented each time a successful (matching) CAN packet is received.
Logix Format	The Data Type to be used for the generated Logix tags. Only applicable when the module is configured as an EtherNet/IP Target.

Table 3.4 – CAN Receive parameters

3.5.2. CAN SEND

The CAN Send configuration is used to determine which CAN Bus packets to send and when.

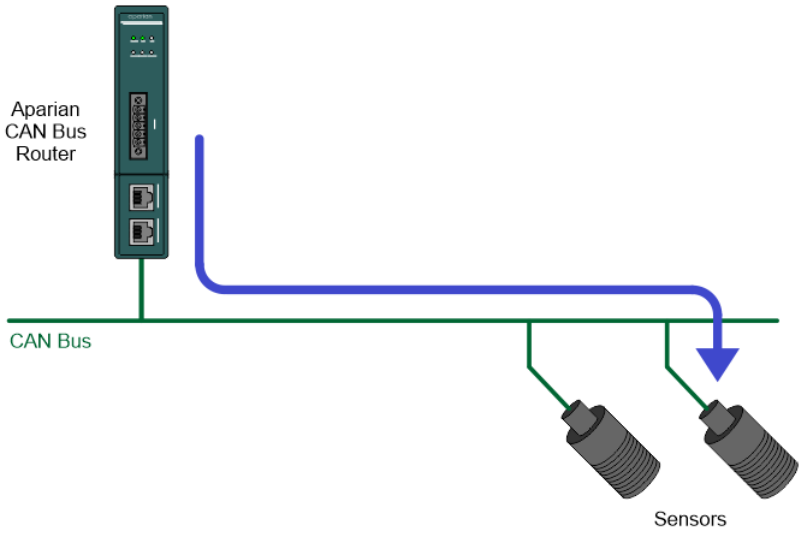


Figure 3.20 – CAN Send data flow

The CAN Send configuration window is opened by either double clicking on the module in the tree, or right-clicking the module and selecting ***Configuration*** and then selecting the ***CAN Send*** tab.

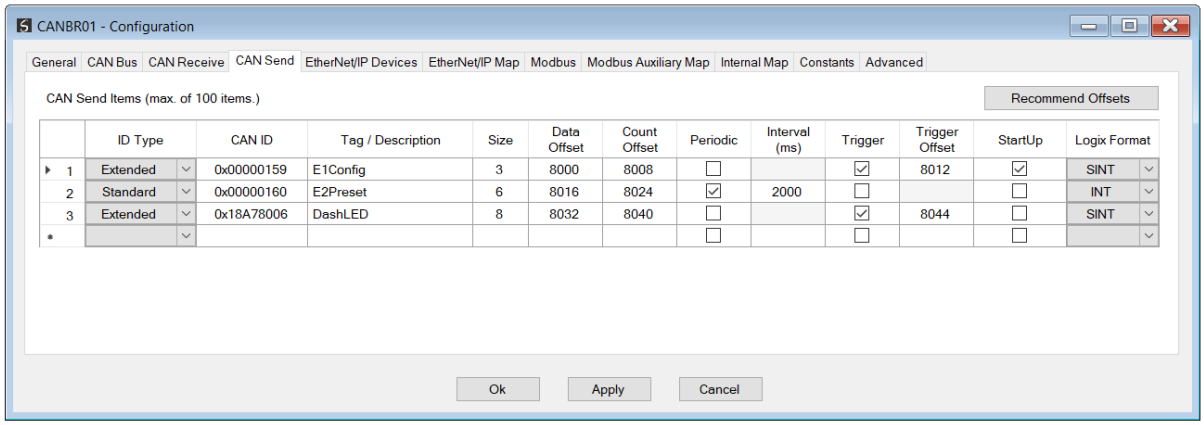


Figure 3.18 – CAN Send Configuration

The **CAN Send** configuration consists of the following parameters:

Parameter	Description
ID Type	The CAN ID Type, either: • Standard – (11-bit CAN Header) • Extended – (29-bit CAN Header)
CAN ID	The CAN ID of the transmitted CAN packet.
Tag / Description	A user defined Tag or Description string for this item. The maximum length of the string is 32 characters.

	Note: When configured as an EtherNet/IP Target, this field will be used in the generation of the Logix tags and must therefore comply with the Logix tag naming rules. (No extended characters or control characters etc.)
Size	The size (in bytes) of the transmitted CAN packet payload. (0-8)
Data Offset	The offset in the Internal Data Space from where the transmitted CAN data will be copied.
Count Offset	The offset in the Internal Data Space where the CAN item transaction count will be copied. (4 bytes). This count is incremented each time this CAN packet is transmitted.
Periodic	Enables Periodic transmission. When enabled, the configured CAN packet will be transmitted periodically, at the rate specified by the ***Interval***.
Interval (ms)	The minimum interval (milliseconds) between transmissions of this CAN packet.
Trigger	Enables Triggered transmission. When enabled, the configured CAN packet will be transmitted each time the trigger value (located at the ***Trigger Offset***) changes.
Trigger Offset	The offset in the Internal Data Space where the CAN item trigger value is located. (4 bytes). Each time the value of the trigger changes, the CAN packet will be transmitted.
StartUp	Enables transmission on module start-up. When enabled, the configured CAN packet will be transmitted when the module boots-up
Logix Format	The Data Type to be used for the generated Logix tags. Only applicable when the module is configured as an EtherNet/IP Target.

Table 3.4 – CAN Send parameters

3.6. PRIMARY INTERFACE

The CAN Bus Router module supports four different modes for the Primary Interface.

3.6.1. ETHERNET/IP TARGET

A controller (e.g. Logix controller) can own the CAN Bus Router over EtherNet/IP using up to 4 Class 1 EtherNet/IP connections when the Primary Interface is set to EtherNet/IP target. This will allow the CAN Bus Router to exchange data with the controller using the input and output assembly of the Class 1 EtherNet/IP connections. Data from the CAN Send and CAN Receive maps, can be mapped to the Logix controller over EtherNet/IP.

The user will need to add the CAN Bus Router to the Logix IO tree under an EtherNet/IP bridge (e.g. 1756-EN2TR) or Ethernet Logix controller (e.g. 1756-L85E).

3.6.1.1. STUDIO / LOGIX 5000 CONFIGURATION

A. ADD MODULE TO ETHERNET/IP I/O CONFIGURATION

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

NOTE: Logix version 21 and newer supports EDS AOPs.

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

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

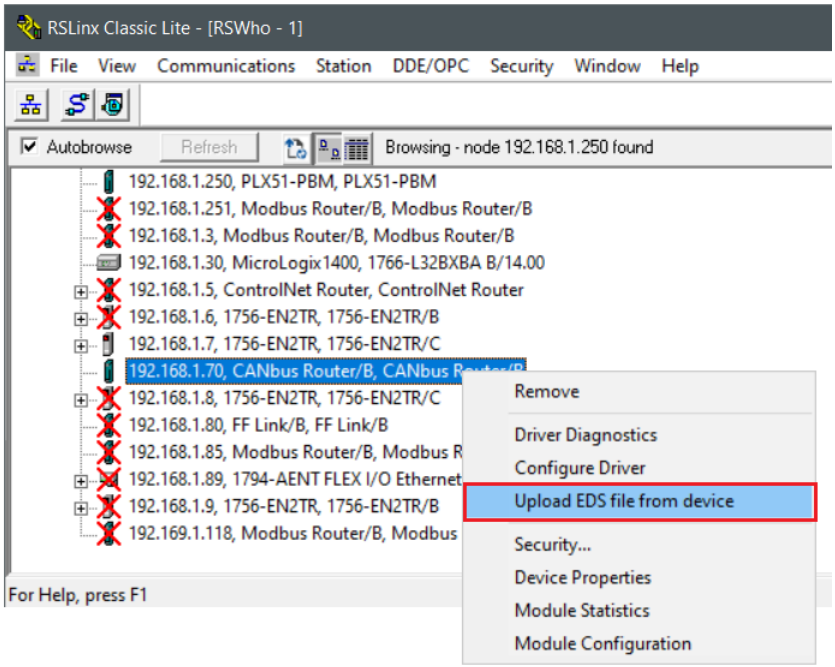


Figure 3.19 – EDS file upload from CAN Bus Router

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

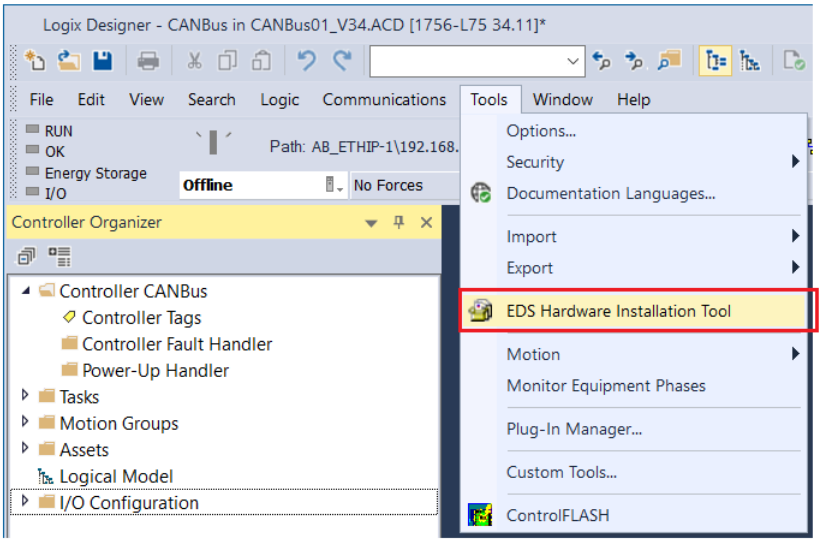


Figure 3.20 - EDS Hardware Installation Utility

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

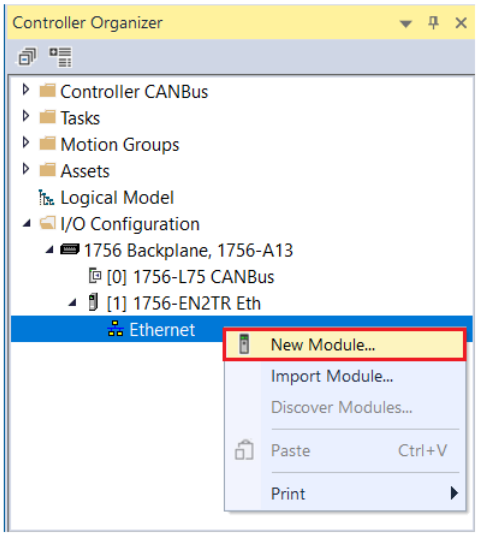


Figure 3.21 – Adding a module

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

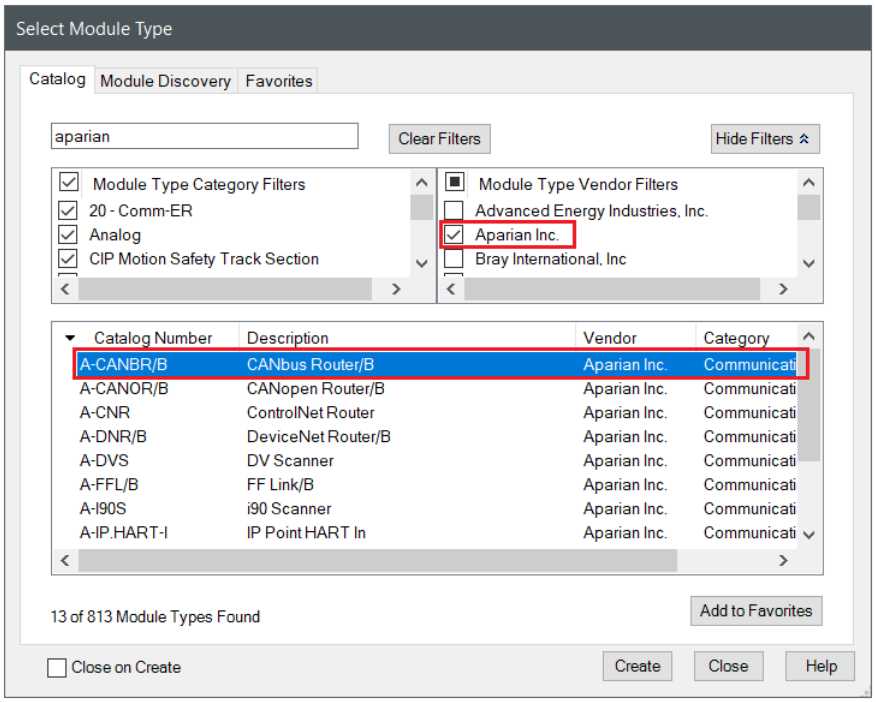


Figure 3.22 – Selecting the module

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

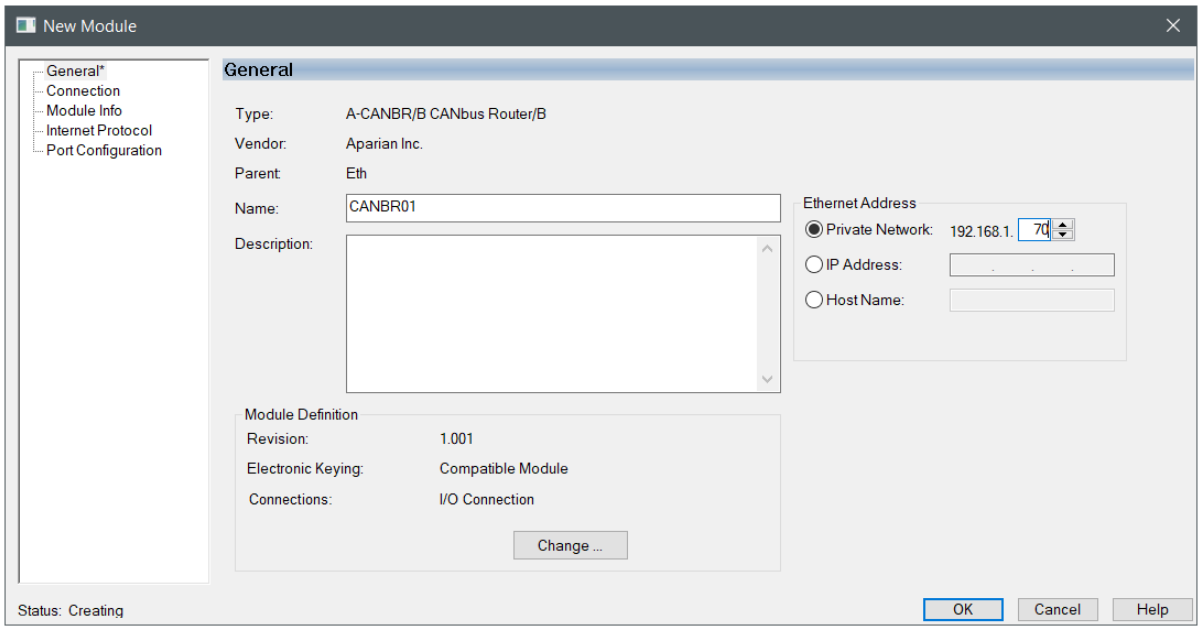


Figure 3.23 – Module instantiation

The CAN Bus Router supports up to 4 class 1 EtherNet/IP connections. The user will need to **ensure that the number of connections configured in the General tab of the module configuration (Slate) matches the selected connection count in Logix (Studio 5000)**.

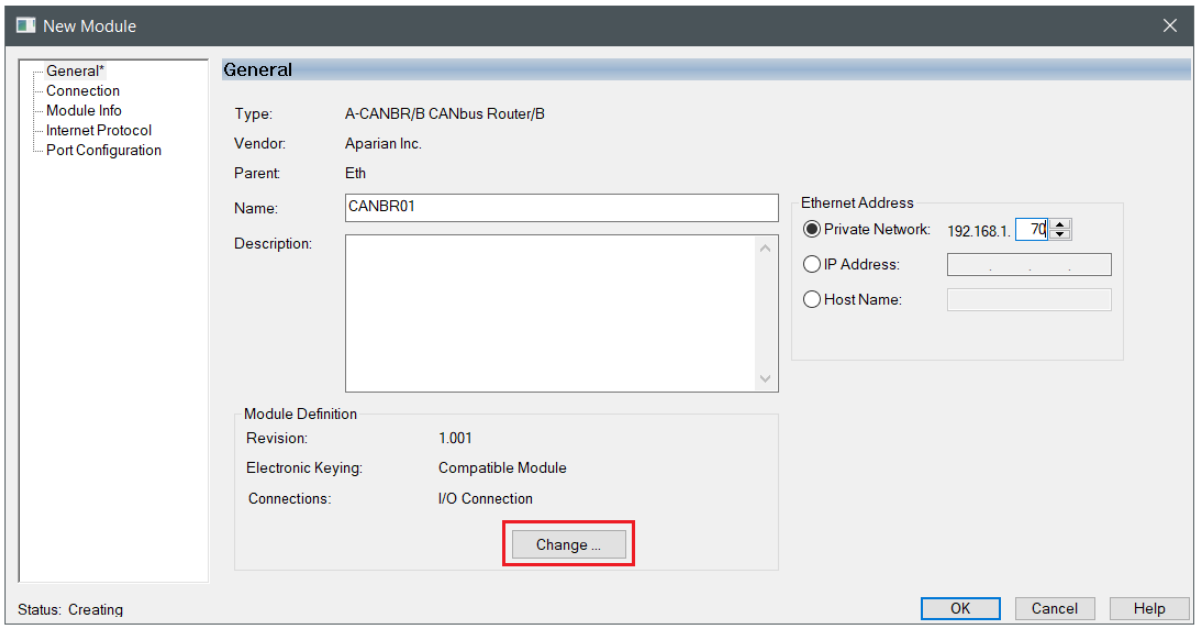


Figure 3.24 – Change number of IO Connections

Next the user will need to select the number of connections required.

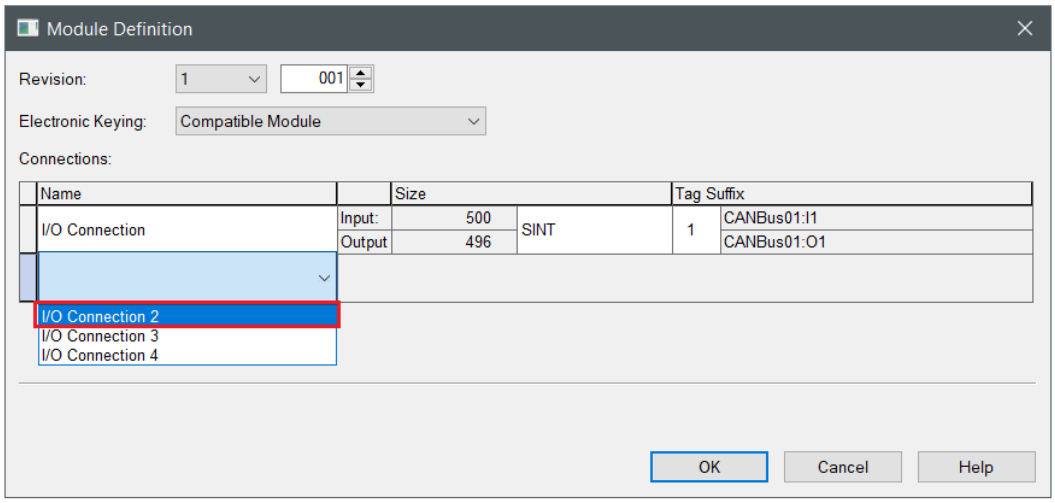


Figure 3.25 – Selection of IO Connections

Now the CAN Bus Router module will be in the Logix IO tree.

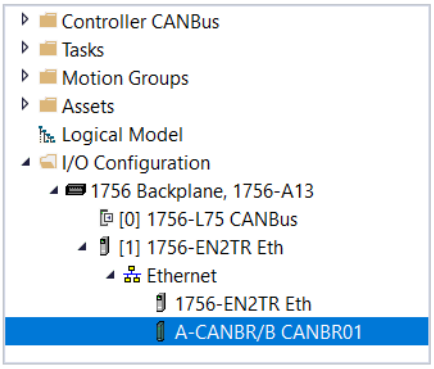


Figure 3.26 – Logix IO tree

The Module Defined Data Types will automatically be created during the instantiation process. These module defined tags will need to be copied to and from meaningful structures.

B. LOGIX MAPPING

Slate will generate the required UDTs and Routines (based on the Internal Map) to map the required CAN Send and Receive items. The user will need to select the ***Recommend*** button in the Internal Mapping to auto populate the Internal Mapping which can then be used to generate the L5X file for Logix mapping, routines, and UDTs

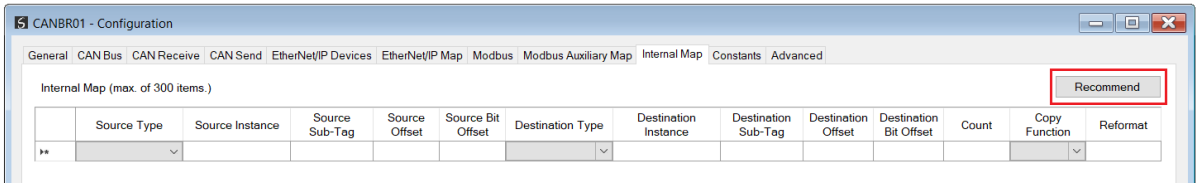


Figure 3.27 – Internal Mapping Recommend

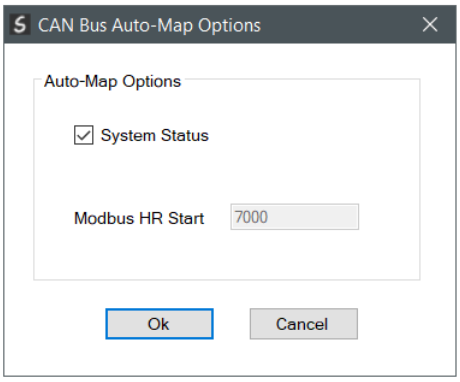


Figure 3.28 – Auto-Map Options popup

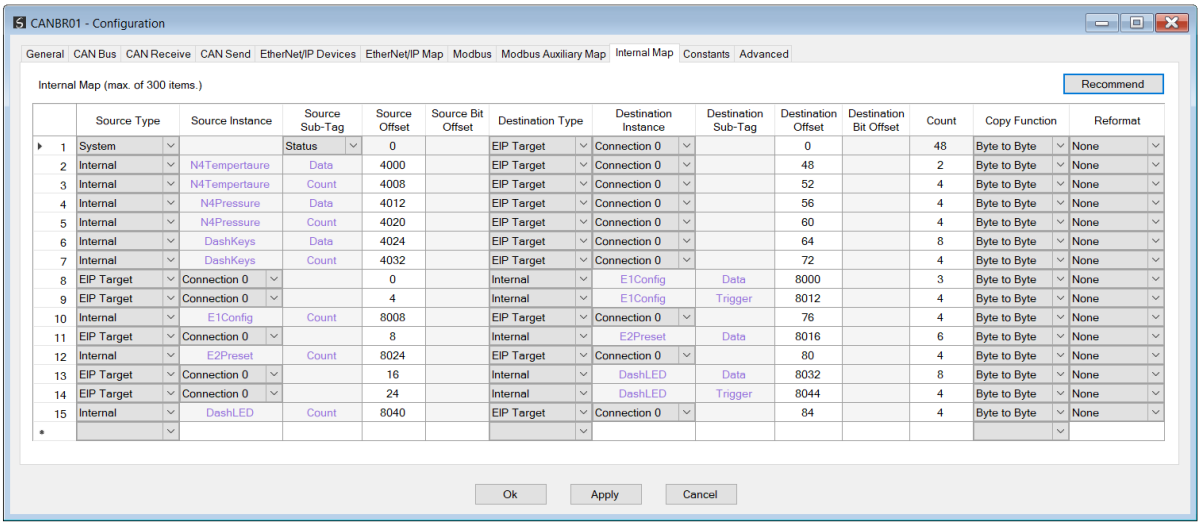


Figure 3.29 – Internal Mapping Auto Populated

The user can then generate the required Logix and UDTs by right-clicking on the module in Slate and selecting the ***Generate Logix L5X*** option.

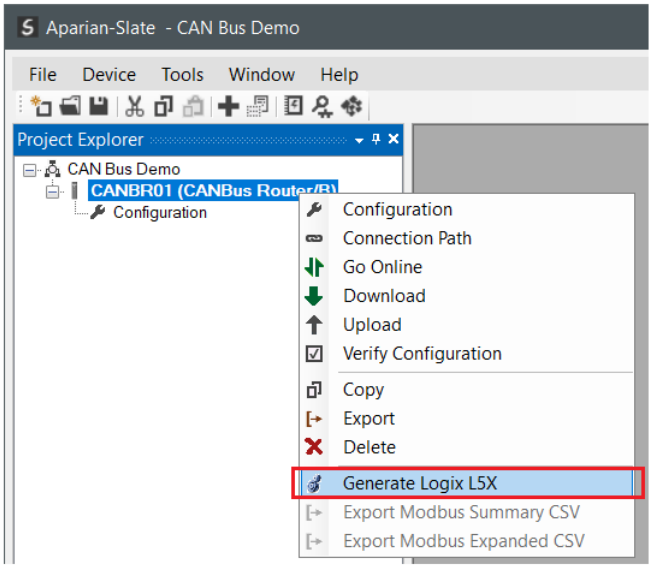


Figure 3.30 – Selecting Generate Logix L5X

The user will then be prompted to select a suitable file name and path for the L5X file.

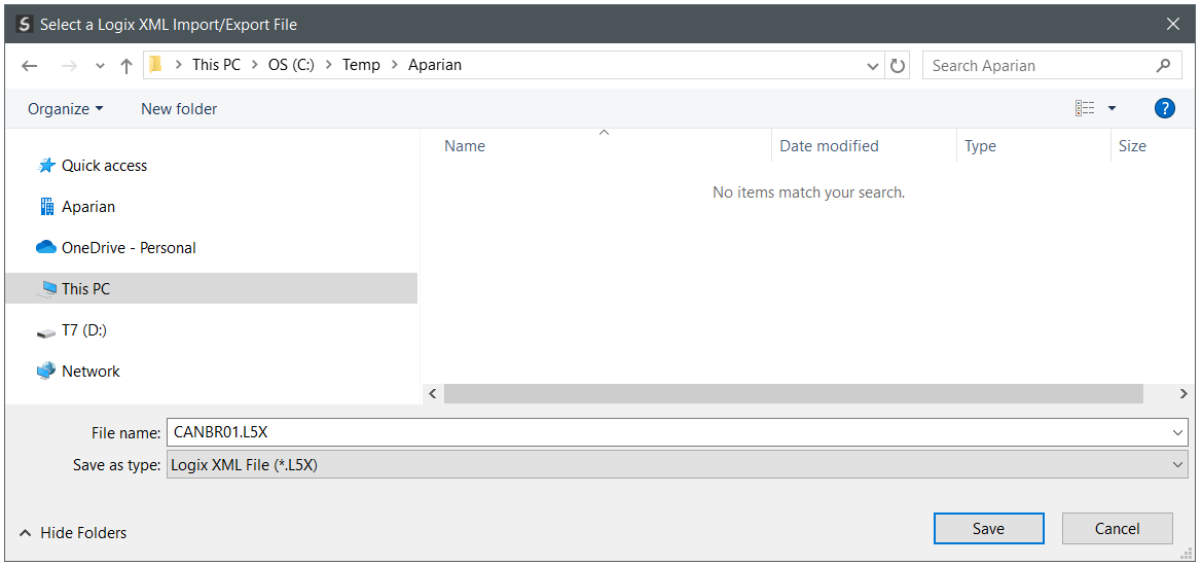


Figure 3.31 – Selecting the Logix L5X file name

This L5X file can now be imported into the Studio 5000 project by right-clicking on a suitable ***Program*** and selecting ***Add***, and then ***Import Routine***.

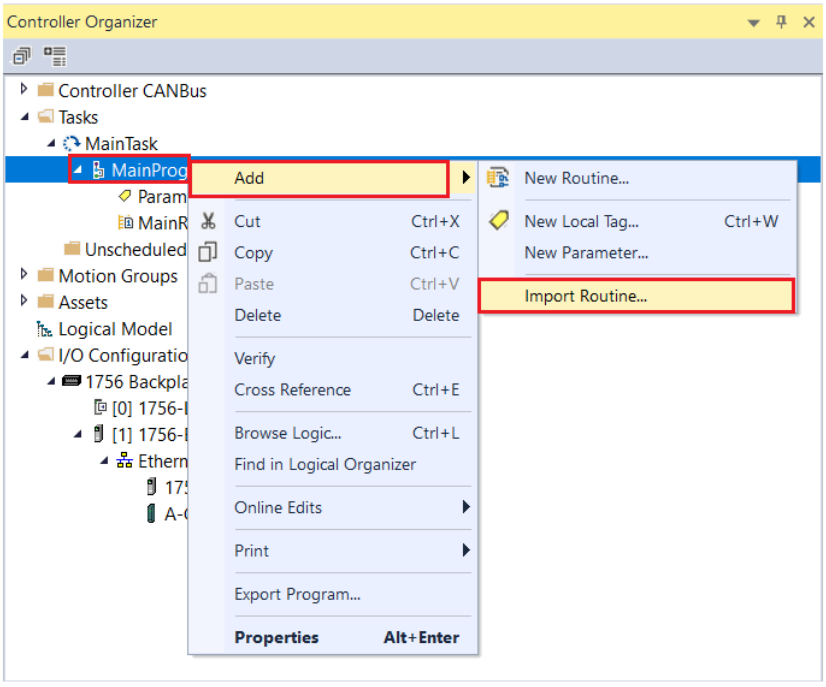


Figure 3.32 – Importing the L5X file into Studio 5000

In the file open dialog select the previously created L5X file and accept the import by pressing ***Ok***.

The import will create the following:

- Mapping Routine
- Multiple UDT (User-Defined Data Types)
- Multiple Controller Tags

Since the imported mapping routine is not a Main Routine, it will need to be called from the current Main Routine.



Figure 3.33 – Calling the mapping routine

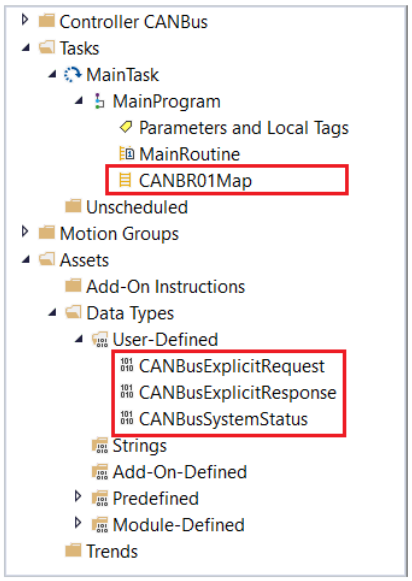


Figure 3.34 – Imported Logix Objects

A number of CAN Bus Router specific (UDT) tags are created. The System tag displays the status and diagnostics information of the CAN Bus Router.

Name	Style	Data Type	Description
CANB01System		CANBusSystemStatus	
CANB01System.ConfigValid	Decimal	BOOL	Configuration Valid
CANB01System.EIPOriginatorCommsOk	Decimal	BOOL	EtherNet/IP Originator 0=Fail, 1=Ok
CANB01System.ModbusOnline	Decimal	BOOL	0=Offline, 1=Online
CANB01System.EIPOwned	Decimal	BOOL	EtherNet/IP Target: 0=Not-Owned, 1=Owned
CANB01System.PowerMainConnector	Decimal	BOOL	Main Power: 0=Off 1=On
CANB01System.PowerCANConnector	Decimal	BOOL	CAN Power: 0=Off 1=On
CANB01System.ControllerRun	Decimal	BOOL	Controller Mode: 0=Program, 1=Run
CANB01System.NTPOk	Decimal	BOOL	NTP Status: 0=Fail, 1=Ok
CANB01System.CANBusOk	Decimal	BOOL	CAN Bus Status: 0=Fail, 1=Ok
CANB01System.ConfigCRC	Hex	INT	Configuration Checksum
CANB01System.TransactionRate	Decimal	INT	Transactions per second
CANB01System.CANBaud	Decimal	SINT	Current CAN Baud: 0=10k, 1=20k, 2=50k, 3=125k, 4=250k, 5=500k, 6=800k, 7...
CANB01System.Temperature	Float	REAL	Module Temperature (deg C)
CANB01System.CANRxPacketCount	Decimal	DINT	CAN Rx Packet Count
CANB01System.CANTxPacketCount	Decimal	DINT	CAN Tx Packet Count
CANB01System.CANCRCErrors	Decimal	DINT	CAN CRC Errors
CANB01System.CANBitErrors	Decimal	DINT	CAN Bit Errors
CANB01System.CANStuffErrors	Decimal	DINT	CAN Stuff Errors
CANB01System.CANBusOffCount	Decimal	DINT	CAN Bus-Off Events
CANB01System.CANAckErrors	Decimal	DINT	CAN Acknowledgment Errors

Figure 3.35 – System Status tag

For each CAN Receive item, there are two tags created. One for the received CAN ***Data***, and one for the item ***Count***.

The data type of the ***Data*** tag is as per the item's configured ***Logix Format***.

The ***Count*** (DINT) is incremented each time the configured CAN item is received.

Name	Value	Style	Data Type
CANBR01DashKeys_Count	2134	Decimal	DINT
CANBR01DashKeys_Data	{...}	Decimal	SINT[8]
CANBR01DashKeys_Data[0]	0	Decimal	SINT
CANBR01DashKeys_Data[1]	0	Decimal	SINT
CANBR01DashKeys_Data[2]	0	Decimal	SINT
CANBR01DashKeys_Data[3]	0	Decimal	SINT
CANBR01DashKeys_Data[4]	0	Decimal	SINT
CANBR01DashKeys_Data[5]	0	Decimal	SINT
CANBR01DashKeys_Data[6]	0	Decimal	SINT
CANBR01DashKeys_Data[7]	0	Decimal	SINT

Figure 3.36 – CAN Receive specific tags

For each CAN Send item, there are two or three tags created. One for the transmitted CAN ***Data***, one for the item ***Count***, and if enabled, one for the ***Trigger***.

The data type of the ***Data*** tag is as per the item's configured ***Logix Format***.

The ***Count*** (DINT) is incremented each time the configured CAN item is transmitted.

The ***Trigger*** (DINT) can be used to manually trigger the CAN item transmission. Any change to this value will trigger a transmission.

Name	Value	Style	Data Type
CANBR01DashLED_Count	423	Decimal	DINT
CANBR01DashLED_Data	{...}	Decimal	SINT[8]
CANBR01DashLED_Data[0]	0	Decimal	SINT
CANBR01DashLED_Data[1]	0	Decimal	SINT
CANBR01DashLED_Data[2]	0	Decimal	SINT
CANBR01DashLED_Data[3]	0	Decimal	SINT
CANBR01DashLED_Data[4]	0	Decimal	SINT
CANBR01DashLED_Data[5]	0	Decimal	SINT
CANBR01DashLED_Data[6]	0	Decimal	SINT
CANBR01DashLED_Data[7]	0	Decimal	SINT
CANBR01DashLED_Trigger	10	Decimal	DINT

Figure 3.37 – CAN Send specific tags

3.6.1.2. INTERNAL DATA SPACE MAPPING

When the module is operating as an EtherNet/IP Target, the data from the originator device (e.g. Logix Controller) can be mapped to the CAN Bus interface using the Internal Map. The Internal Map configuration window is opened by either double clicking on the module in the tree or right-clicking the module and selecting *Configuration* and selecting the *Internal Map* tab.

A. IDS COPY – ETHERNET/IP TARGET SOURCE

When copying data from a connection originator (e.g. the output assembly from the Logix Controller) to the CAN Bus interface, the source type needs to be EIP Target.

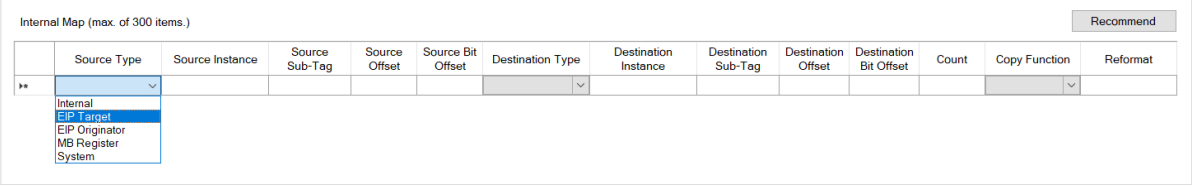


Figure 3.38 – IDS Copy – EtherNet/IP Target Source Type

The source instance will be the connection number, which can be connection 0 to 3, based on the number of connections configured. The Source Offset is the offset in the EtherNet/IP

output assembly from where the data must be copied. The Count is the number of **bytes** that will be copied.

See the Internal Data Space Mapping section for more information regarding the operation.

B. IDS COPY – ETHERNET/IP TARGET DESTINATION

When copying data from the CAN Bus interface to the EtherNet/IP Target input assembly, the destination type needs to be EIP Target.

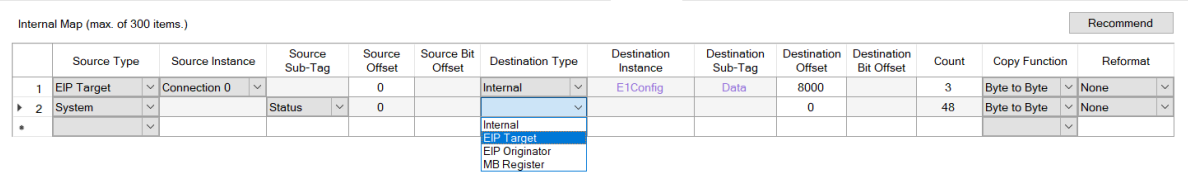


Figure 3.39 – IDS Copy – EtherNet/IP Target Destination Type

The destination instance will be the connection number, which can be connection 0 to 3, based on the number of connections configured. The Destination Offset is the offset of the EtherNet/IP input assembly from where the data must be copied. The Count is the number of **bytes** that will be copied.

3.6.2. MODBUS SERVER AND CLIENT

3.6.2.1. MODBUS SERVER

The CAN Bus Router can operate as a Modbus Server for Modbus TCP, RTU232, and RTU485 simultaneously. A Modbus Client can read and write to the full Modbus Register range in the CAN Bus Router.

3.6.2.2. MODBUS CLIENT

The CAN Bus Router can operate as a Modbus Client for Modbus TCP, RTU232, and RTU485 simultaneously. In addition to the relevant Modbus Parameters shown below, the user will also need to configure the Modbus Auxiliary Map. This map will allow the user to configure various read and write functions to external Modbus Registers, to and from the internal Modbus registers.

3.6.2.3. MODBUS CONFIGURATION

For both Modbus Server and Modbus Client, the user will need to configure the relevant Modbus Parameters as shown 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*** and selecting the ***Modbus*** tab.

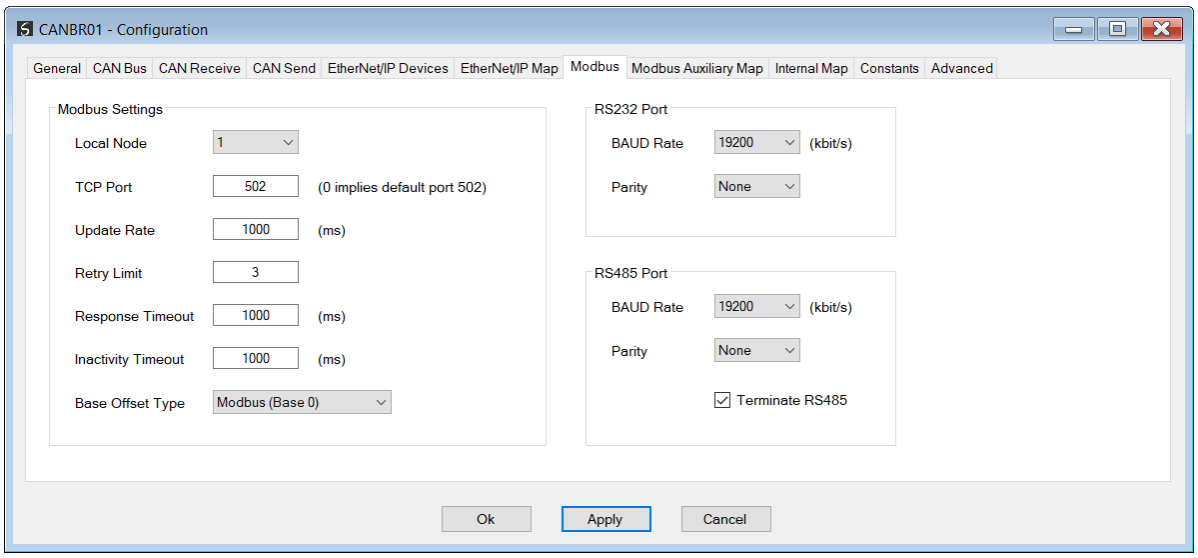


Figure 3.40 – Modbus Configuration

The Modbus Communication configuration consists of the following parameters:

Parameter	Description
Local Node	The Modbus Node address assigned to the CAN Bus Router.
TCP Port	The TCP port to be used for the Modbus communication. By default, the module will use the standard TCP port 502.
Update Rate	The period (in milliseconds) between Client requests to the target Modbus device. **(Modbus Client mode only)**
Retry Limit	The number of successive Modbus request retries before the request is set to have failed. **(Modbus Client mode only)**
Response Timeout	The time (in milliseconds) the module will wait for a valid Modbus response. **(Modbus Client mode only)**
Inactivity Timeout	The amount of time during which no Modbus requests have been received before the CAN Bus Router indicates that the connection to the Modbus Client is no longer active. **(Modbus Server mode only)**
Base Offset Type	**Modbus (Base 0)** Conventional Modbus addressing where the first address is 0. **PLC (Base 1)**

	PLC addressing, where the first address is 1.
RS232 Port	
BAUD Rate	The RS232 serial port's BAUD rate. (Modbus RTU232)
Parity	The RS232 serial port's Parity configuration. (Modbus RTU232)
RS485 Port	
BAUD Rate	The RS485 serial port's BAUD rate. (Modbus RTU485)
Parity	The RS485 serial port's Parity configuration. (Modbus RTU485)
Terminate RS485	Enables the on-board 125Ω RS485 terminating resistor.

Table 3.3 – Modbus parameters

The Modbus Node Number will need to be configured in the parameters above to allow a Modbus Client to access the CAN Bus Router as a Modbus Server device.

3.6.2.4. INTERNAL DATA SPACE MAPPING

When the module is operating as a Modbus Server, the data from the Modbus Registers (used to exchange data with the Modbus Client) can be mapped to the CAN Bus interface using the Internal Map. The Internal Map configuration window is opened by either double clicking on the module in the tree or right-clicking the module and selecting ***Configuration*** and selecting the ***Internal Map*** tab.

NOTE: The user can select the *Recommend* button in the Internal Map to auto map the CAN Bus status and data to recommended Modbus Registers.

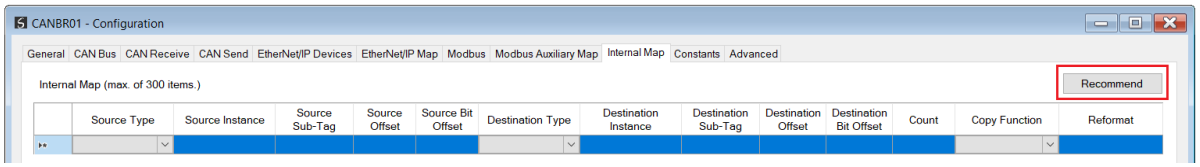


Figure 3.41 – Modbus Server – Internal Mapping Recommend

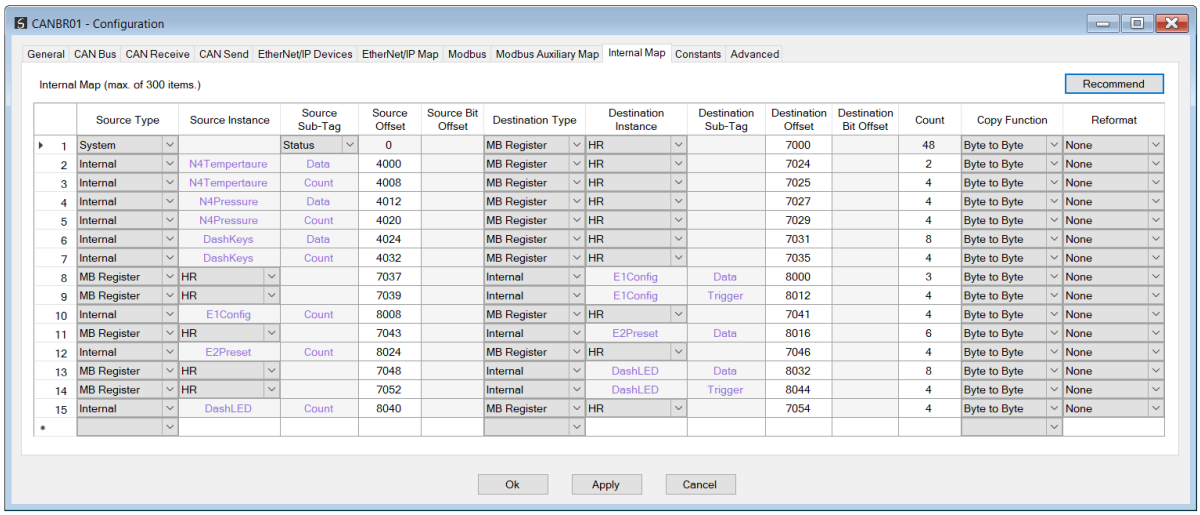


Figure 3.42 – Modbus Server – Internal Mapping Updated

A. IDS COPY – MODBUS SOURCE

When copying Modbus data the source type needs to be MB Register.

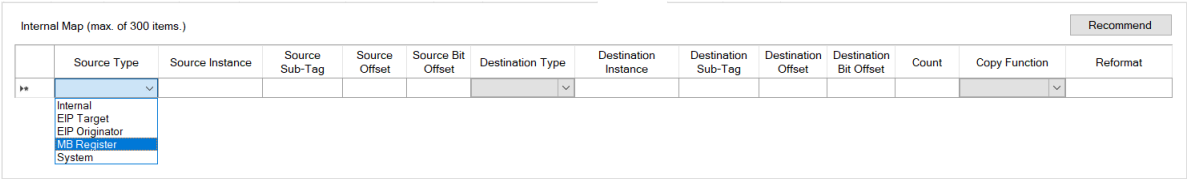


Figure 3.43 – IDS Copy - Modbus Source Type

The source instance will be the Modbus register type required.

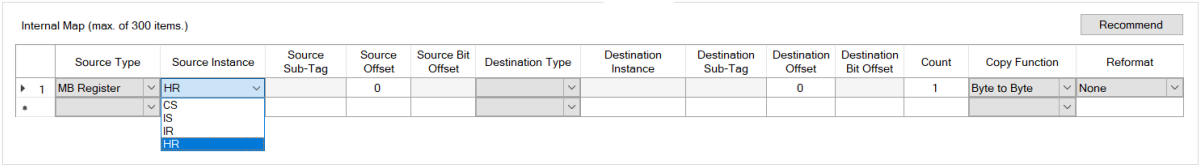


Figure 3.44 – IDS Copy - Modbus Source Instance

The Source Offset is the Modbus Register offset from where the data must be copied. The Count is the number of **bytes** that will be copied. See the Internal Data Space Mapping section for more information regarding the operation.

B. IDS COPY – MODBUS DESTINATION

When copying data to a Modbus Register, the destination type needs to be MB Register.

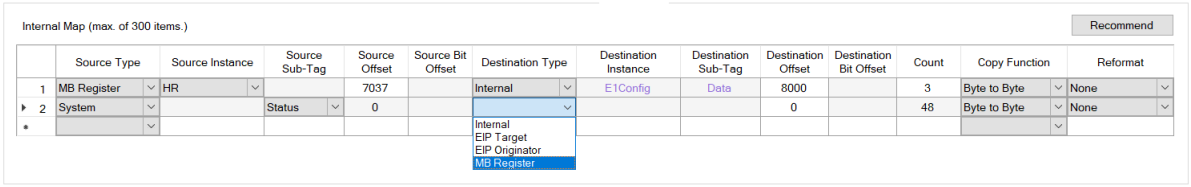


Figure 3.45 – IDS Copy - Modbus Destination Type

The destination instance will be the Modbus register type required.

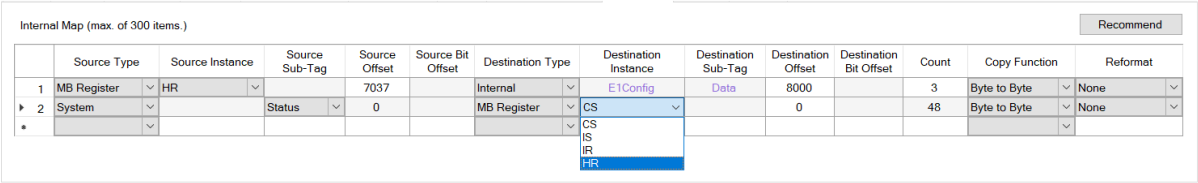


Figure 3.46 – IDS Copy - Modbus Destination Instance

The Destination Offset is the Modbus Register offset to where the data must be copied. The Count is the number of **bytes** that will be copied. See the Internal Data Space Mapping section for more information regarding the operation.

3.6.2.5. MODBUS AUXILIARY MAP

The ***Modbus Auxiliary Map*** configuration is shown in the figure below. The Modbus configuration is only applicable when the module has a Modbus Client operating interface. Up to 100 mapping items can be configured. Within the 100 mapping items, a maximum of 20 Modbus TCP Server devices can be configured.

The ***Modbus Auxiliary Map*** will be executed in a sequential manner and a mapped item will be executed at the configured ***Update Rate*** in the Modbus parameters. That is, the ***Update Rate*** is the time (milliseconds) between two successive mapped item executions.

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

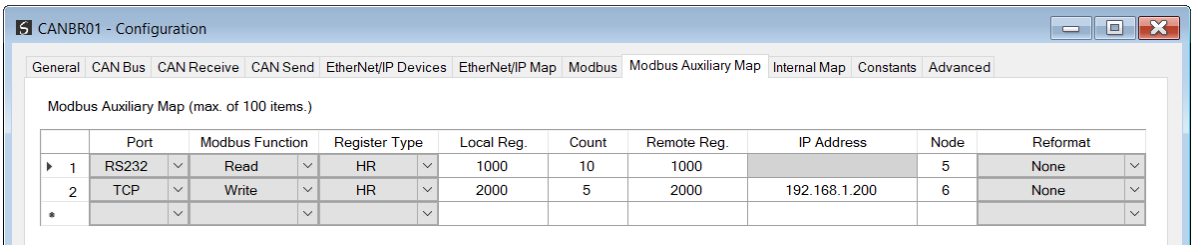


Figure 3.47 – Modbus Auxiliary Map Configuration

The ***Modbus Auxiliary Map*** configuration consists of the following parameters:

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

Table 3.4 – Modbus Auxiliary Map parameters

3.6.3. EtherNet/IP Originator

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

3.6.3.1. ETHERNET/IP CLASS 1 DEVICE CONNECTIONS

The CAN Bus Router can establish up to 10 cyclic Class 1 EtherNet/IP connections to EtherNet/IP devices. This can be done by either manually entering the connection data into the ***Connection Parameter*** window, or by importing the configuration from one or more of the following sources:

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

A. MANUAL CONFIGURATION

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

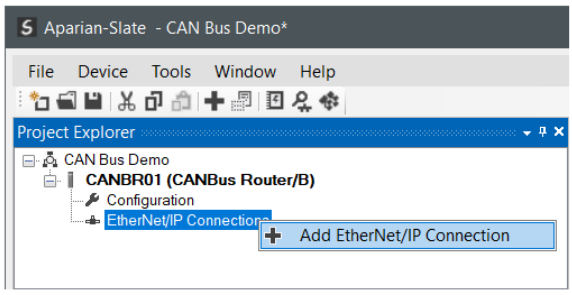


Figure 3.48 – Adding EtherNet/IP Class 1 Connection

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

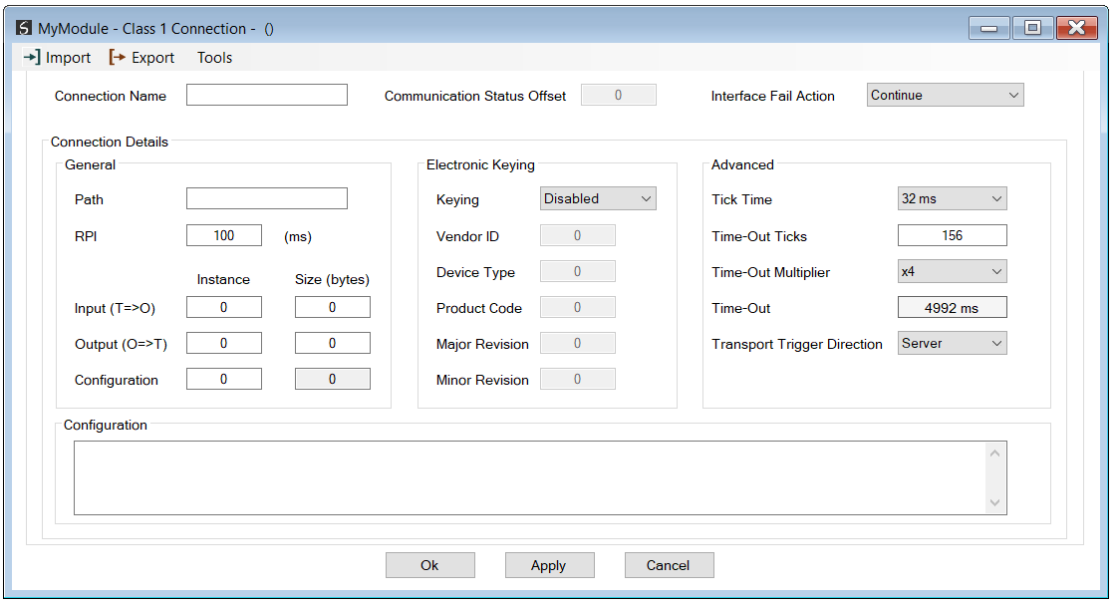


Figure 3.49 – EtherNet/IP Class 1 Connection Parameters



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

Parameter	Description
Connection Name	The instance name given to the Class 1 Connection.
Interface Fail Action	When the CAN Bus communication has failed, the EtherNet/IP IO can be configured to either keep the connection running as is, change the connection status to program mode, or force the connection offline. This will allow the EtherNet/IP device to go into a pre-determined state when the CAN Bus communication fails.
General	
Path	The path to the target device. If the device is an Ethernet device, then this will just be the IP address of the module. If the device is, for example, a module in a backplane or via an adapter, then the user will need to enter the IP address of the bridge or adapter followed by the backplane port (for example 1) and the slot number of the device. Each item is separated by a comma. As an example, to connect to an Allen-Bradley Flex module (via the Flex Adapter at IP address 192.168.1.100) that is in slot 2 of the Flex backplane, the user will need to enter the following path: 192.168.1.100,1,2 (IP address, port (backplane), slot).
RPI	The requested packet interval (RPI) is the rate in milliseconds at which the data will be sent from the originator to the target and vice versa.
Input (T=>O) – Instance	The instance of the input assembly.
Input (T=>O) – Size (bytes)	The size in bytes of the input assembly.
Output (O=>T) – Instance	The instance of the output assembly.
Output (O=>T) – Size (bytes)	The size in bytes of the output assembly.
Configuration – Instance	The instance of the configuration assembly.
Configuration – Size (bytes)	The size in bytes of the configuration assembly. **NOTE:** This is a read-only value and will be equal to the number of bytes entered into the configuration window below.
Electronic Keying	
Keying	Electronic Keying can be used to ensure that the target device is the correct device type. **Disabled** Keying is not enabled, and no key information will be sent in the connection establishment. **Compatible**

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

Table 3.5 – EtherNet/IP Class 1 Connection Parameters

B. IMPORT FROM ONLINE CONTROLLER

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

PREPARATION

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

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

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

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

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

IMPORT CONNECTION PARAMETERS

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

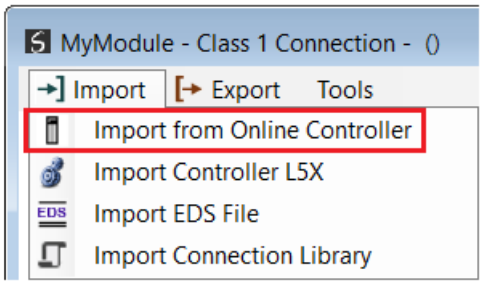


Figure 3.50 – Import from Online Controller

The Import Connection Parameters form will open.

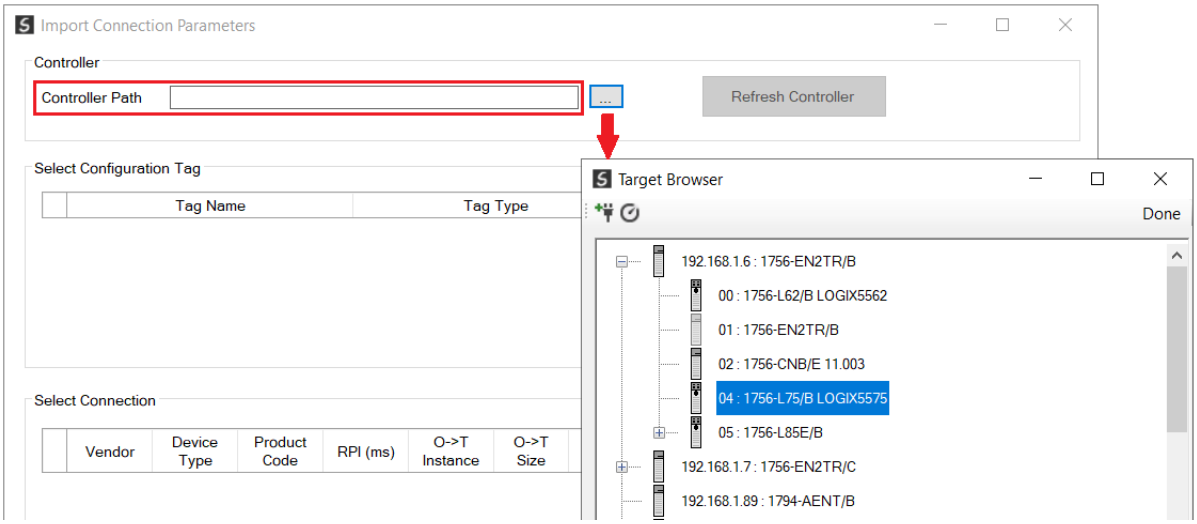


Figure 3.51 – Import Connection Parameters – Controller Path

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

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

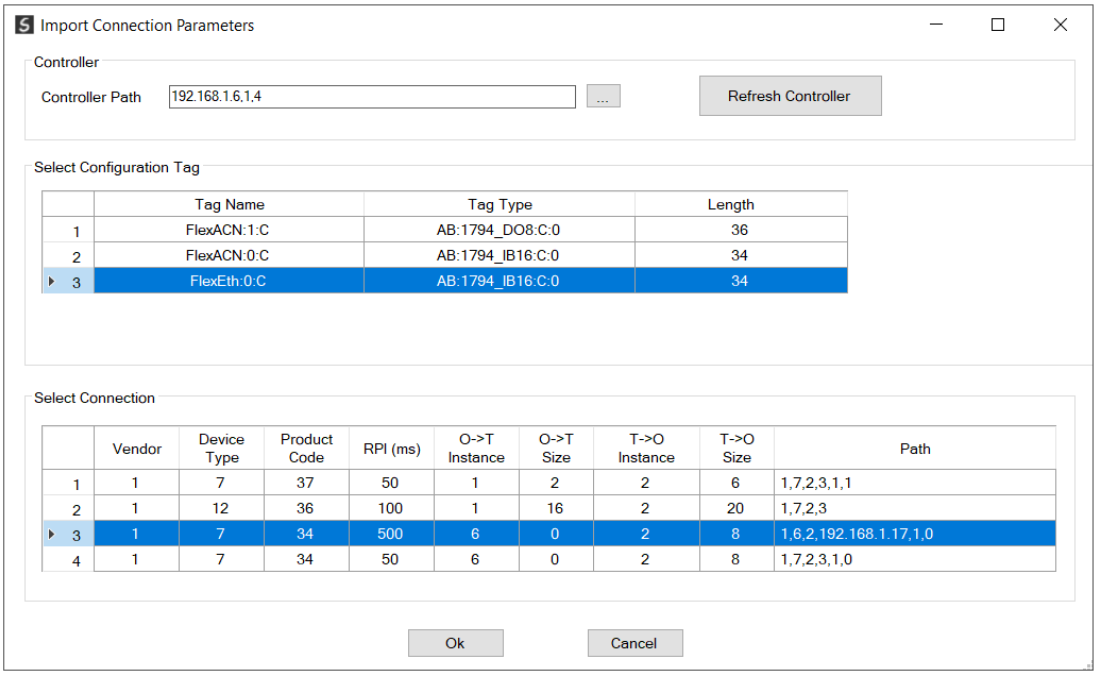


Figure 3.52 – Import Connection Parameters – Select Connection

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

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

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

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

C. IMPORT FROM CONTROLLER L5X FILE

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

PREPARATION

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

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

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

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

IMPORT L5X FILE

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

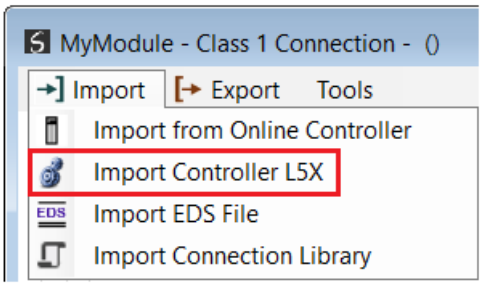


Figure 3.53 – Import from Controller L5X

The Import Connection Parameters form will open.

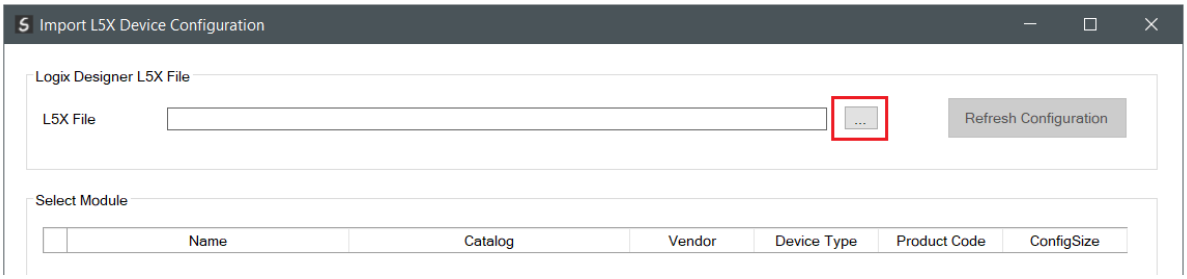


Figure 3.54 – Import L5X Device Configuration – Select L5X

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

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

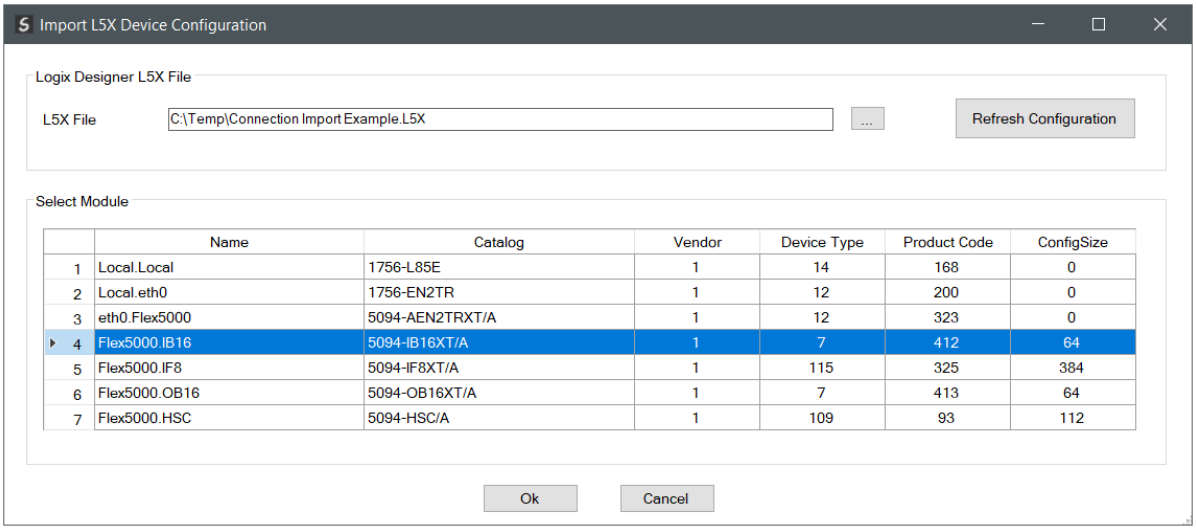


Figure 3.55 – Import L5X Device Configuration

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

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

D. IMPORT EDS FILE

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

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

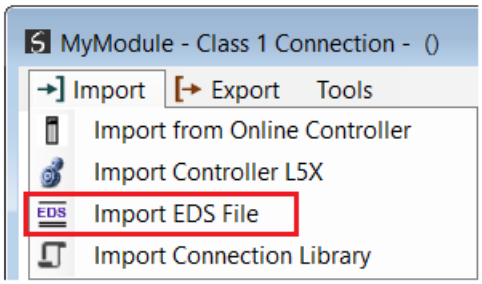


Figure 3.56 – Import EDS File

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

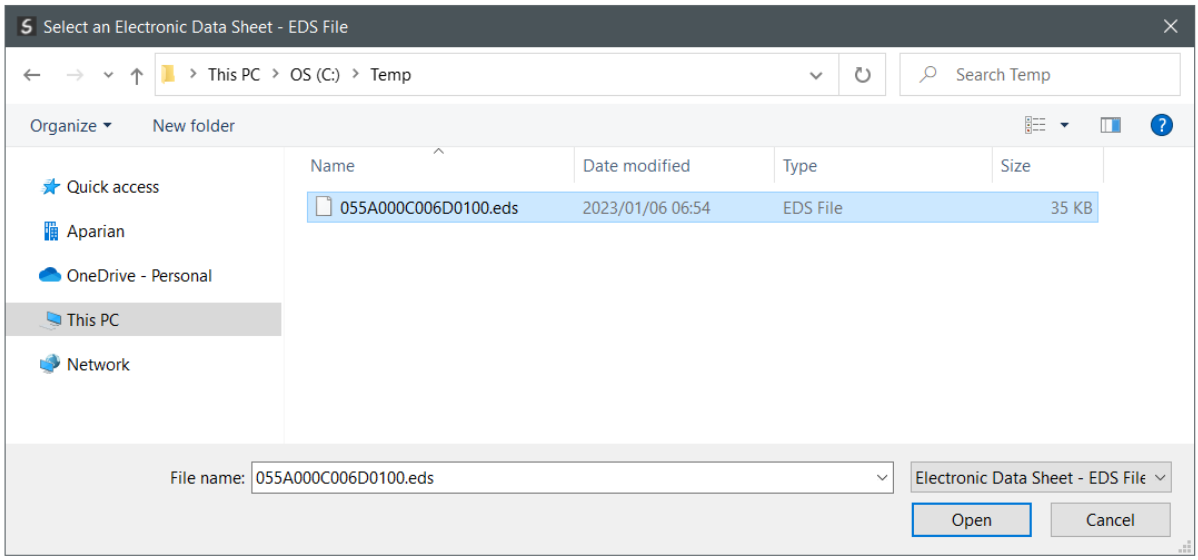


Figure 3.57 – Browse to EDS File

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

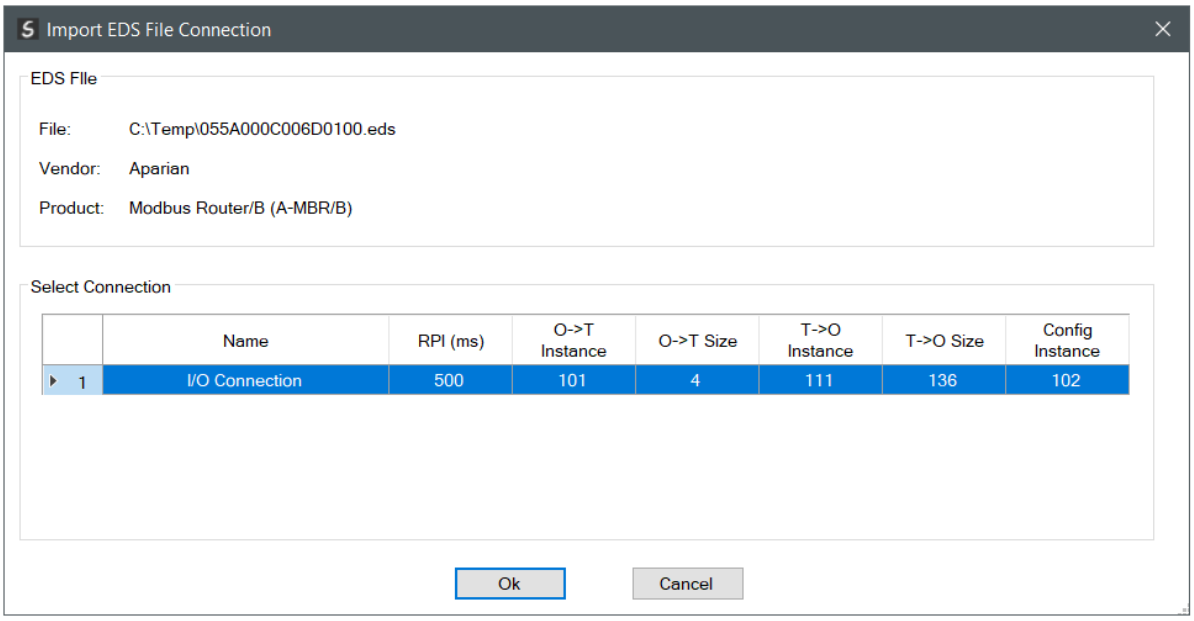


Figure 3.58 – Select Connection

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

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

E. IMPORT CONNECTION LIBRARY

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

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

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

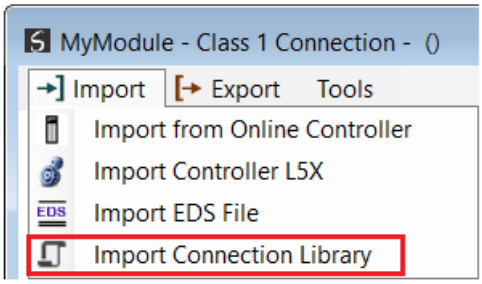


Figure 3.59 – Import Connection Library File

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

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

EXPORT LIBRARY FILE

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

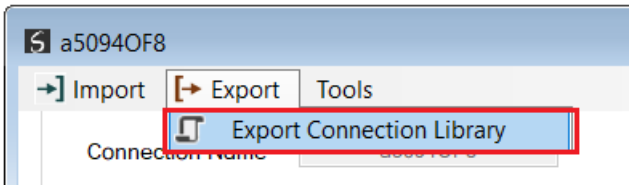


Figure 3.60 – Export Connection Library File

3.6.3.2. ETHERNET/IP EXPLICIT MESSAGE DEVICE CONNECTIONS

Up to 10 EtherNet/IP devices can be added for explicit messaging. The user will need to add each device as explained in the EtherNet/IP Devices section below. Once the EtherNet/IP devices have been added the user can then configure the required mapping for the EtherNet/IP Explicit messaging as shown in EtherNet/IP Map section below.

A. ETHERNET/IP DEVICES

This tab is enabled when the Primary Interface is set to *EtherNet/IP Originator*.

The EtherNet/IP Devices configuration is shown in the figure below. Up to 10 EtherNet/IP devices can be configured with up to 50 EtherNet/IP mapped items allowing for either explicit EtherNet/IP Class 3 or Unconnected Messaging (UCMM) to any of the 10 configured devices. The data from each EtherNet/IP device is written to, or read from, an Internal Data Space with a size of 100Kbytes. See the *Explicit EtherNet/IP Messaging* section for more details.

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

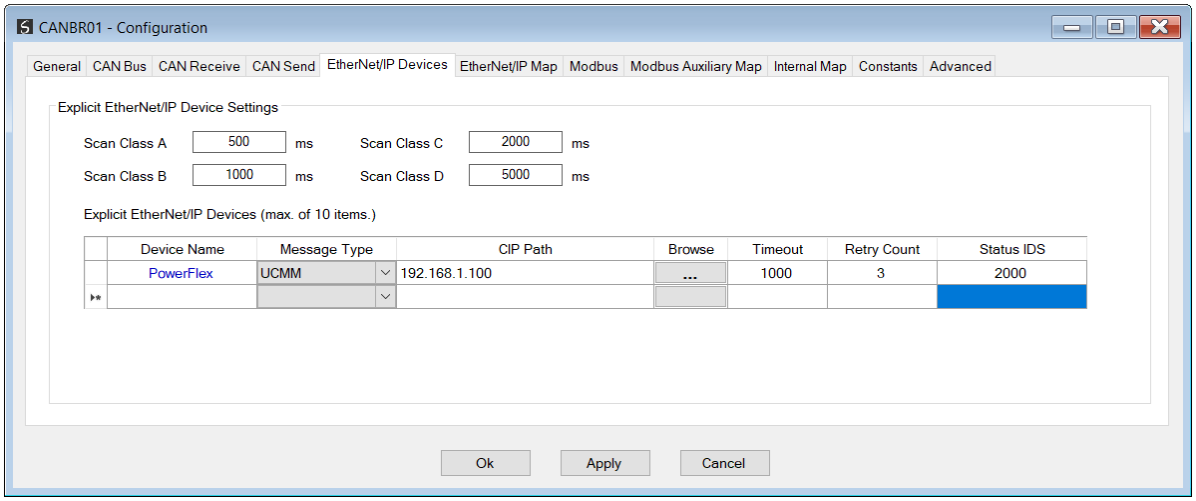


Figure 3.61 – EtherNet/IP Devices - Configuration

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

Parameter	Description
Scan Class A, B, C, D	The configurable update rates (in milliseconds) for each scan class in the EtherNet/IP Map.

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

Table 3.6 – EtherNet/IP Devices configuration parameters

B. ETHERNET/IP MAP

This tab is enabled when the Primary Interface is set to *EtherNet/IP Originator*.

The EtherNet/IP Map configuration is shown in the figure below. Up to 50 EtherNet/IP mapped items, either explicit EtherNet/IP Class 3 or Unconnected Messaging (UCMM) to any of the 10 pre-configured devices can be configured. The data from each EtherNet/IP device is written to or read from Internal Data Space with a size of 100Kbytes. See the *Explicit EtherNet/IP Messaging* section for more details.

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

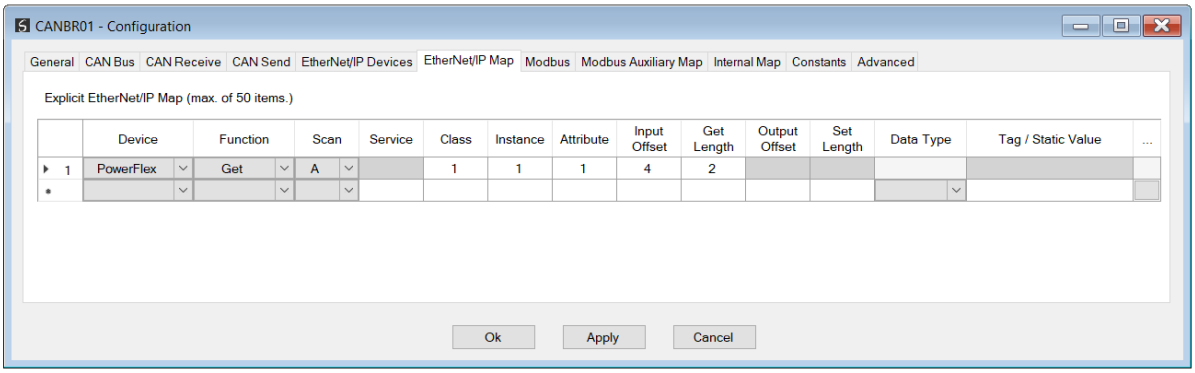


Figure 3.62 – EtherNet/IP Map configuration

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

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

	When using a Logix controller as a EtherNet/IP Device, the CAN Bus Router module can read a Logix tag from the target Logix controller using Class 3 or UCMM messaging. The value from the tag will be saved at the configured Input Offset. **Write Tag** When using a Logix controller as a EtherNet/IP Device, the CAN Bus Router module can write to Logix tag from the target Logix controller using Class 3 or UCMM messaging. The value from the tag will be read from the configured Output Offset.
Scan	The user can select Scan Class **A**, **B**, **C** or **D** (which was configured in the EtherNet/IP Devices tab). The specific mapped item will then be executed at that configured scan class rate. The user can also select the **S** class which means that the mapped item will only execute once when communication to the target device is established. If the target device goes offline, then the mapped items with this class will be re-armed, and resent when communication is re-established.
Service	The custom CIP service/function which is only available when the ***Custom*** function has been selected.
Class, Instance, Attribute	The CIP class, instance, and attribute of the request message to be sent.
Input Offset	The location in the Internal Data Space where the received data will be written. This will only be available for ***Get*** and ***Custom*** functions.
Get Length	The length of the data to be received. If the number of bytes received is more than the Get Length, then the data will **not** be written to the Internal Data Space.  **NOTE:** When the function is Logix Read, then the Get Length will be the number of **elements** of the configured data type and **not** the byte count. This will only be available for ***Get, Custom*** and ***Read Tag*** functions.
Output Offset	The location in the Internal Data Space from where the data to be written to the target device will be read. This will only be available for ***Set*** and ***Custom*** functions.
Set Length	The length of the data to be written.  **NOTE:** When the function is Logix Write, then the Set Length will be the number of **elements** of the configured data type and **not** the byte count. This will only be available for ***Set, Custom*** and ***Write Tag*** functions.
Data Type	The data type of the Static Value. This will only be available for ***Set Static*** function.

Tag / Static Value	The value to be written to the target device when the ***Set Static*** function has been selected. **Note**: When using the SINT Array data type, the values must be entered as space-delimited hex values. For example: 05 34 2E A1

Table 3.7 – EtherNet/IP Map configuration parameters

3.6.3.3. INTERNAL DATA SPACE MAPPING

When the module is operating as a EtherNet/IP Originator, the data from the EtherNet/IP IO devices can be mapped to the CAN Bus interface (via Internal Data Space) using the Internal Map. The Internal Map configuration window is opened by either double clicking on the module in the tree or right-clicking the module and selecting *Configuration* and selecting the *Internal Map* tab.

A. IDS COPY – ETHERNET/IP ORIGINATOR SOURCE

When copying data from an EtherNet/IP IO device the source type needs to be EIP Originator.

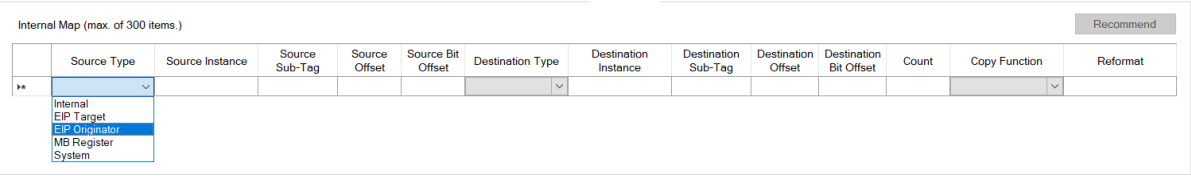


Figure 3.63 – IDS Copy – EtherNet/IP Originator Source Type

The source instance will be one of the EtherNet/IP IO devices added to the EtherNet/IP IO tree in Slate.

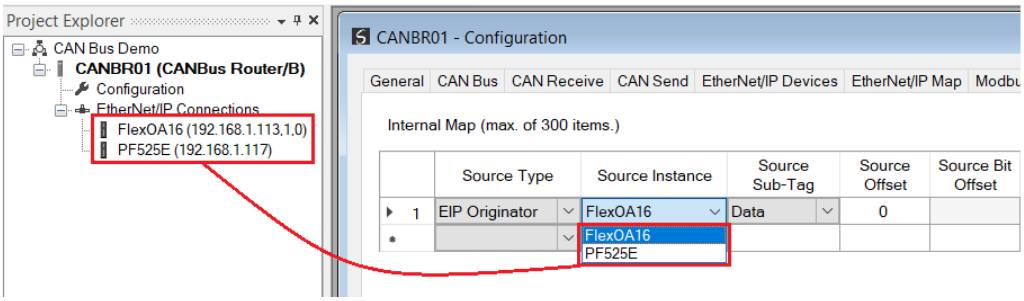


Figure 3.64 – IDS Copy – EtherNet/IP Originator Source Instance

The Source Offset is the offset in the selected EtherNet/IP device Class 1 **Input** Assembly. The Count is the number of **bytes** that will be copied. See the Internal Data Space Mapping section for more information regarding the operation.

The user can select to copy either the ***Data***, or ***Status***, from the EtherNet/IP connection.

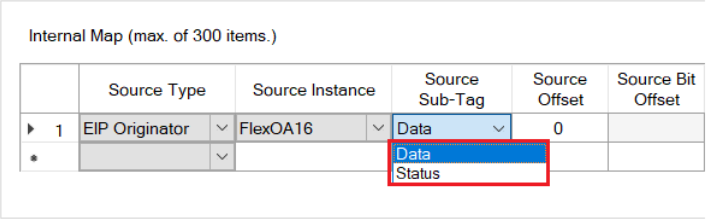


Figure 3.65 – IDS Copy – EtherNet/IP Originator Status

When selecting the ***Status***, option the format of the Status information is as follows:

Parameter	Data Type	Description
EtherNet/IP Originator Connection Status	DINT	Bit 0 – Connection Ok

Table 3.8 – EtherNet/IP Originator Connection Status

B. IDS COPY – ETHERNET/IP TARGET DESTINATION

When copying data to an EtherNet/IP IO device device's **Output** Assembly, the destination type needs to be EIP Originator.

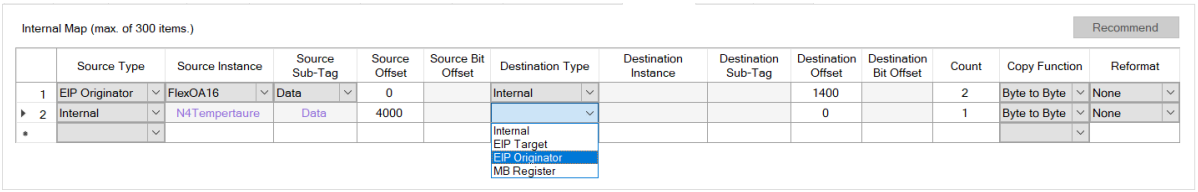


Figure 3.66 – IDS Copy – EtherNet/IP Originator Destination Type

The destination instance will be one of the EtherNet/IP IO devices added to the EtherNet/IP IO tree in Slate.

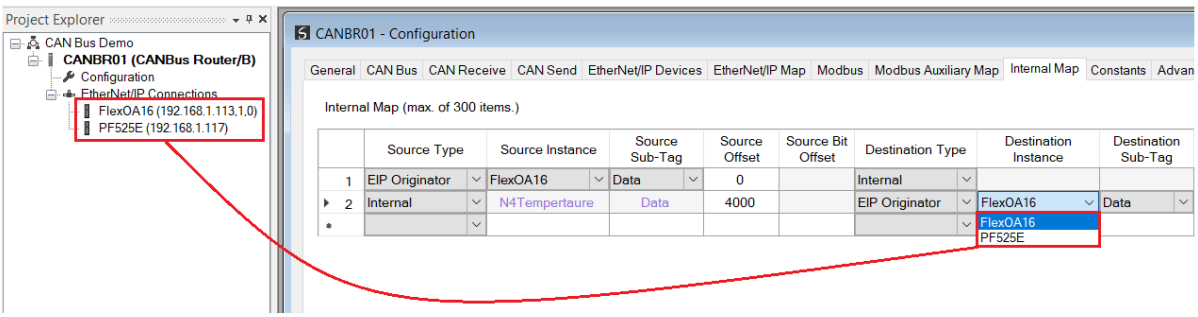


Figure 3.67 – IDS Copy – EtherNet/IP Originator Destination Instance

The Destination Offset is the offset in the selected EtherNet/IP device Class 1 **Output** Assembly. The Count is the number of **bytes** that will be copied. See the Internal Data Space Mapping section for more information regarding the operation.

3.7. INTERNAL DATA SPACE MAP

The internal data map is used to exchange data from the Ethernet interface to the CAN Bus interface and vice versa. Up to 300 items can be mapped. The Internal Map configuration window is opened by either double clicking on the module in the tree, or right-clicking the module and selecting ***Configuration*** and then selecting the ***Internal Map*** tab.

The ***Count*** is the number of bytes that will be copied from the source to the destination. There are four different ***Copy Functions*** that can be used.

Function	Description
Byte to Byte	Each **byte** from the source will be directly copied to each **byte** in the destination.
Byte to Bit	Each byte from the source will be copied to each bit in the destination. If a value greater than zero is read from the **source byte** then a 1 will be written to the **destination bit** address. If a value of zero is read from the **source byte** then a 0 will be written to the **destination bit** address. The destination offset will be the bit offset and the destination address will be incremented by one bit each time.
Bit to Bit	Each **bit** from the source will be directly copied to each **bit** in the destination.
Bit to Byte	Each bit from the source will be copied to each byte in the destination. If a value of one is read from the **source bit** then a 1 will be written to the **destination byte** address. If a value of zero is read from the **source bit** then a 0 will be written to the **destination byte** address. The source offset will be the bit offset and the source address will be incremented by one bit each time.

Table 3.9 – Internal Map Copy functions

The data in the destination source can also be reformatted. The reformat option provides five different reformat options.

NOTE: The reformat option is only available for ***Byte to Byte*** Copy Functions.

Function	Description
None	No reformatting applied (AA BB CC DD)
BB AA	16-bit Byte swap
BB AA DD CC	32-bit Byte Pair Swap

CC DD AA BB	Word Swap
DD CC BB AA	Word and Byte Pair Swap

Table 3.10 – Internal Map Reformat Options

3.7.1. COPY FROM

One of five sources can be selected to copy from:
Internal, EIP Target, EIP Originator, MB Register and System.

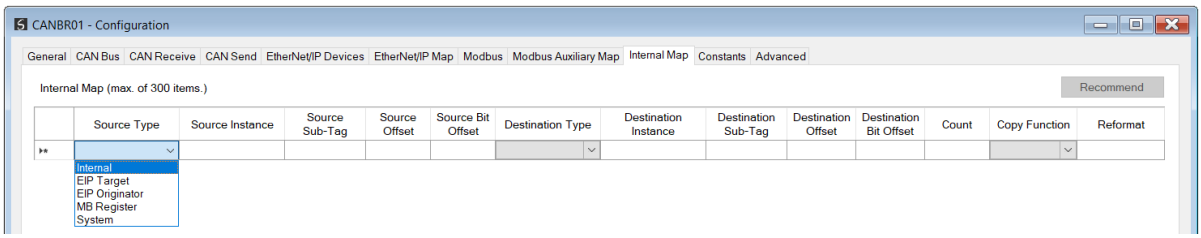


Figure 3.68 – Internal Map – Source Type

3.7.1.1. INTERNAL

When copying data from the internal data space (IDS), the source type needs to be Internal.

	Source Type	Source Instance	Source Sub-Tag	Source Offset	Source Bit Offset
1	Internal			5020	
*					

Figure 3.69 – IDS Copy – Internal Source Type

The source instance is Not Applicable for the internal data space. The Source Offset is the offset in the *Internal Data Space (IDS)* which has a max of 100,000 bytes. The Count is the number of **bytes** that will be copied.

3.7.1.2. EIP TARGET

When copying data from a connection originator (e.g. the output assembly from the Logix Controller) the source type needs to be EIP Target.

	Source Type	Source Instance	Source Sub-Tag	Source Offset	Source Bit Offset
1	EIP Target	Connection 0		0	
*					

Figure 3.70 – IDS Copy – EtherNet/IP Target Source Type

The source instance will be the connection number, which can be connection 0 to 3, based on the number of connections configured. The Source Offset is the offset in the EtherNet/IP output assembly from where the data must be copied. The Count is the number of **bytes** that will be copied.

3.7.1.3. EIP ORIGINATOR

When copying data from an EtherNet/IP IO device, the source type needs to be EIP Originator.

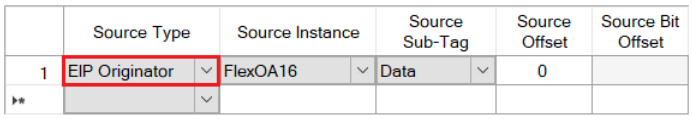


Figure 3.71 – IDS Copy – EtherNet/IP Originator Source Type

The source instance will be one of the EtherNet/IP IO devices added to the EtherNet/IP IO tree in Slate.

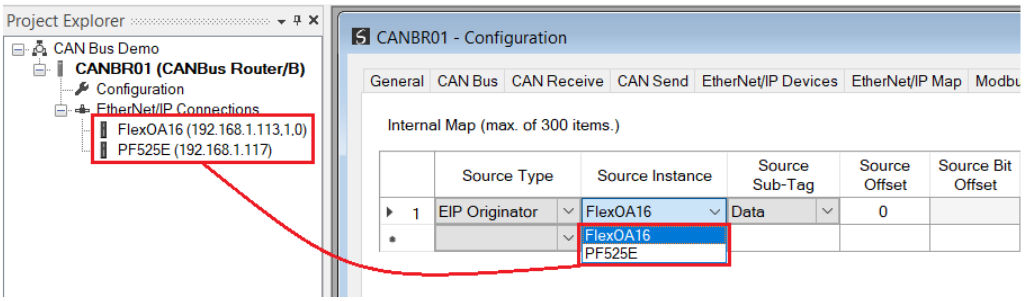


Figure 3.72 – IDS Copy – EtherNet/IP Originator Source Instance

The Source Offset is the offset in the selected EtherNet/IP device Class 1 **Input** Assembly. The Count is the number of **bytes** that will be copied.

The user can select to copy the data from the EtherNet/IP connection or the status.

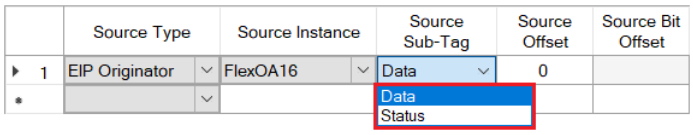


Figure 3.73 – IDS Copy – EtherNet/IP Originator Status

When selecting the status the format of the Status information is shown below:

Parameter	Data Type	Description
EtherNet/IP Originator Connection Status	DINT	Bit 0 – Connection Ok

Table 3.11 – EtherNet/IP Originator Connection Status

3.7.1.4. MODBUS REGISTER

When copying from Modbus data the source type needs to be MB Register.

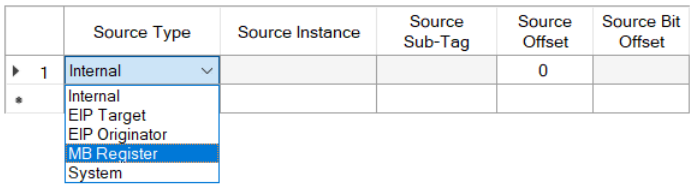


Figure 3.74 – IDS Copy - Modbus Source Type

The source instance will be the Modbus register type required.

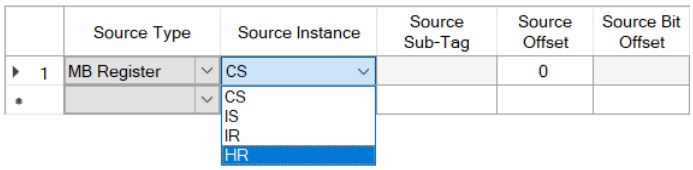


Figure 3.75 – IDS Copy - Modbus Source Instance

The Source Offset is the Modbus Register offset from where the data must be copied. The Count is the number of **bytes** that will be copied.

3.7.1.5. SYSTEM

When copying system information, the source type needs to be System.

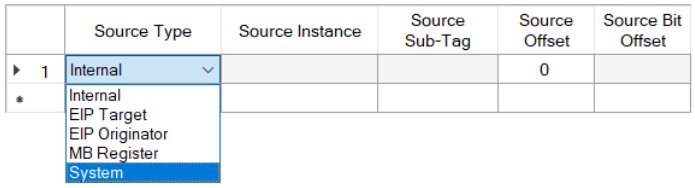


Figure 3.76 – IDS Copy – System Information

The module's System information has the following format.

Parameter	Data Type	Description
Status	DINT	Module Status. Bit 0 – Module Config Valid Bit 1 – EtherNet/IP Originator Comms Ok Bit 2 – Modbus Comms Ok Bit 3 – EtherNet/IP Target Comms Ok Bit 4 – Power is applied to the bottom connector Bit 5 – Power is applied to the front connector Bit 6 – Controller in Run Mode Bit 7 – NTP Ok Bit 8 – CAN Bus Activity Ok
Config CRC	INT	Configuration CRC Checksum
Transaction Rate	INT	Number of CAN Bus transactions executed every second.
Current CAN BAUD Rate	SINT	Current CAN BAUD Rate 0=10k, 1=20k, 2=50k, 3=125k, 4=250k, 5=500k, 6=800k, 7=1M
Reserved	SINT[3]	Reserved
Temperature	REAL	The internal temperature of the module.
Rx CAN Packet Count	DINT	The number of CAN Bus packets received.
Tx CAN Packet Count	DINT	The number of CAN Bus packets sent.
CAN CRC Errors	DINT	The number of received packets where the packet checksum does not match the calculated packet checksum. This implies one or more bits in the frame have been corrupted. May indicate an intermittent CAN cable connection or induced electrical noise.
CAN Bit Errors	DINT	The number of transmitted bits where the transmitted bit state does not match the instantaneous read-back state. This may indicate that another device is transmitting at the same time, or one of the CAN lines shorted to power, shorted together, or incorrectly termination.
CAN Stuff Errors	DINT	The number of frames received where the required inserted (opposite) bit was not received after 5 identical bits. This may be an indication of bus noise, bad physical cable connection, or a faulty device.
Bus Off	DINT	The number of Bus-Off Events. A node will enter the Bus-Off state when the transmit Error Count exceeds a certain threshold (typically 125). This may indicate a cable break or loss of power causing the Scanner to enter the Bus-Off state.

Ack Error	DINT	The number of transmitted bits that are not read-back and acknowledged by at least one other node. Typically seen when the device is alone on the bus, and there is no other node to acknowledge the frame.
Format Error	DINT	The number of received frames where the fixed format part of a received frame is invalid, or the Frame structure is non-standard. (Frame size/type start delimiter etc.) May indicate an intermittent CAN cable connection, induced electrical noise or a node present with an incorrect BAUD rate.

Table 3.12 – System Information Format

3.7.2. COPY TO

One of four destinations can be selected to copy to:
Internal, EIP Target, EIP Originator and MB Register.

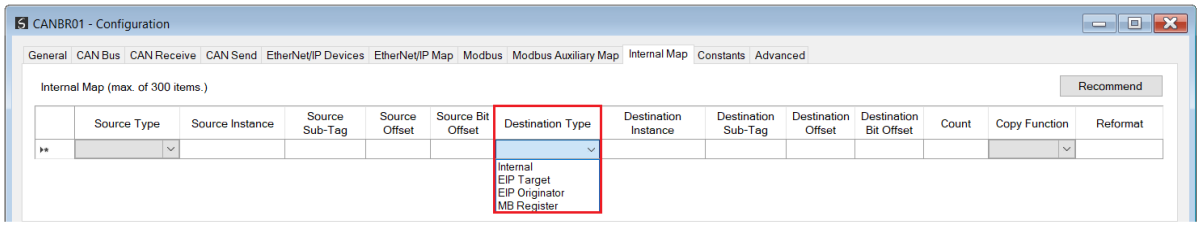


Figure 3.77 – Internal Map – Destination Type

3.7.2.1. INTERNAL

When copying data to the internal data space (IDS), the destination type needs to be Internal.

Destination Type	Destination Instance	Destination Sub-Tag	Destination Offset	Destination Bit Offset	Count	Copy Function	Reformat
Internal			8100		2	Byte to Byte	None

Figure 3.78 – IDS Copy – Internal Source Type

The destination instance is Not Applicable for the internal data space. The Destination Offset is the offset in the *Internal Data Space (IDS)* which has a max of 100,000 bytes. The Count is the number of **bytes** that will be copied.

3.7.2.2. EIP TARGET

When copying data to the EtherNet/IP Target input assembly, the destination type needs to be EIP Target.

Destination Type	Destination Instance	Destination Sub-Tag	Destination Offset	Destination Bit Offset	Count	Copy Function	Reformat
EIP Target	Connection 0		0		2	Byte to Byte	None

Figure 3.79 – IDS Copy – EtherNet/IP Target Destination Type

The destination instance will be the connection number, which can be connection 0 to 3, based on the number of connections configured. The Destination Offset is the offset of the EtherNet/IP input assembly from where the data must be copied. The Count is the number of **bytes** that will be copied.

3.7.2.3. EIP ORIGINATOR

When copying data to an EtherNet/IP IO device's **Output** Assembly, the destination type needs to be EIP Originator.

Destination Type	Destination Instance	Destination Sub-Tag	Destination Offset	Destination Bit Offset	Count	Copy Function	Reformat
EIP Originator	FlexOA16	Data	0		2	Byte to Byte	None

Figure 3.80 – IDS Copy – EtherNet/IP Originator Destination Type

The destination instance will be one of the EtherNet/IP IO devices added to the EtherNet/IP IO tree in Slate.

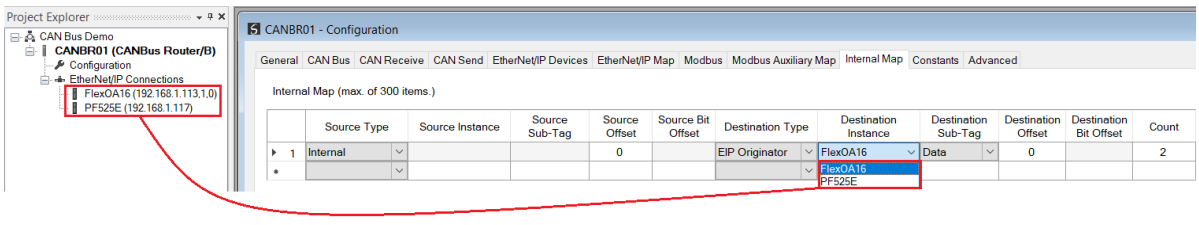


Figure 3.81 – IDS Copy – EtherNet/IP Originator Destination Instance

The Destination Offset is the offset in the selected EtherNet/IP device's Class 1 **Output** Assembly. The Count is the number of **bytes** that will be copied.

3.7.2.4. MODBUS REGISTER

When copying data to a Modbus Register, the destination type needs to be MB Register.

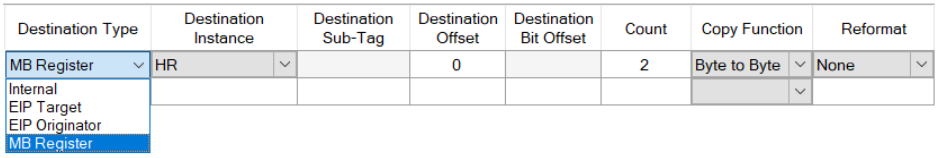


Figure 3.82 – IDS Copy - Modbus Destination Type

The destination instance will be the Modbus register type required.

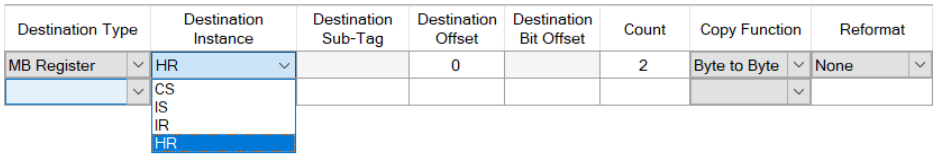


Figure 3.83 – IDS Copy - Modbus Destination Instance

The Destination Offset is the Modbus Register offset to where the data must be copied. The Count is the number of **bytes** that will be copied.

NOTE: For more information regarding the specific Internal Map operation for specific interfaces, see the setup and configuration sections for the various CAN Bus and Primary Interfaces.

3.8. CONSTANTS

The Internal Data Constant configuration provides a mechanism to load the Internal Data Space (IDS) with constant data at module start-up. This is often useful for the pre-population of configuration data for CAN packets, as well as Modbus and EtherNet/IP devices.

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

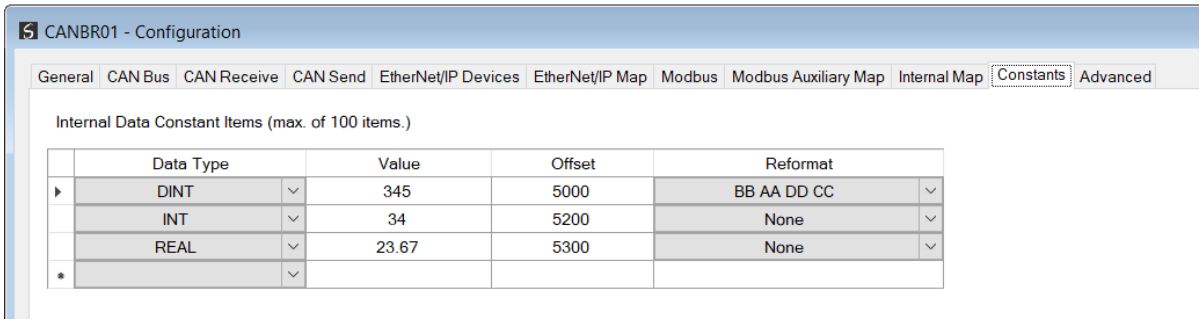


Figure 3.84 – Constants configuration

The ***Constant*** configuration consists of the following parameters:

Parameter	Description
Data Type	The Data Type of the constant item, either: • SINT • INT • DINT • LINT • REAL • USINT • UINT • UDINT
Value	The constant value.
Offset	The offset in the Internal Data Space where the constant value will be copied.
Reformat	Used to specify how the data is formatted before writing to Internal Data Space: • **None** – No reformatting applied. (AA BB or AA BB CC DD). • **BB AA** – 16bit Byte swap • **BB AA DD CC** – 32bit Byte Pair Swap • **CC DD AA BB** – Word Swap • **DD CC BB AA** – Word and Byte Pair Swap

Table 3.13 – Constants configuration parameters

3.9. ADVANCED

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

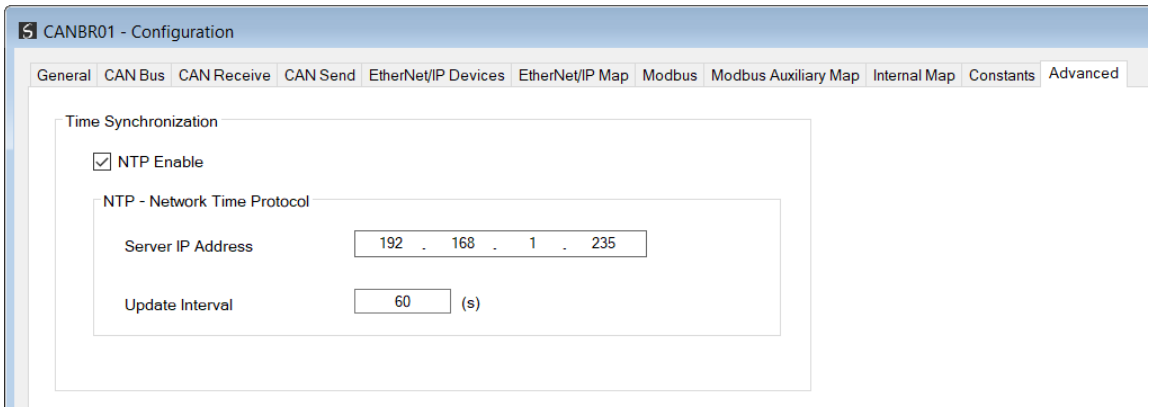


Figure 3.85 – Advanced configuration

The Advanced configuration consists of the following parameters:

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

Table 3.14 – Advanced configuration parameters

3.10. MODULE DOWNLOAD

Once the CAN Bus 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 CAN Bus 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.

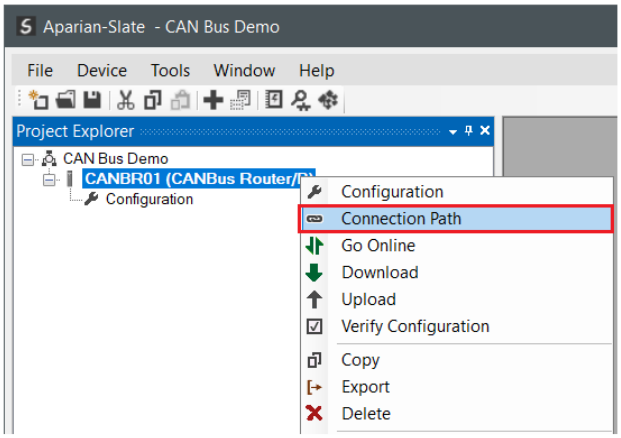


Figure 3.86 - Selecting Connection Path

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

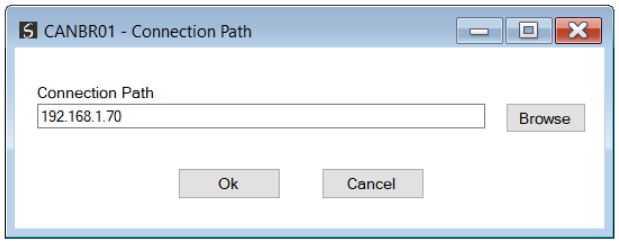


Figure 3.87 - Connection Path

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

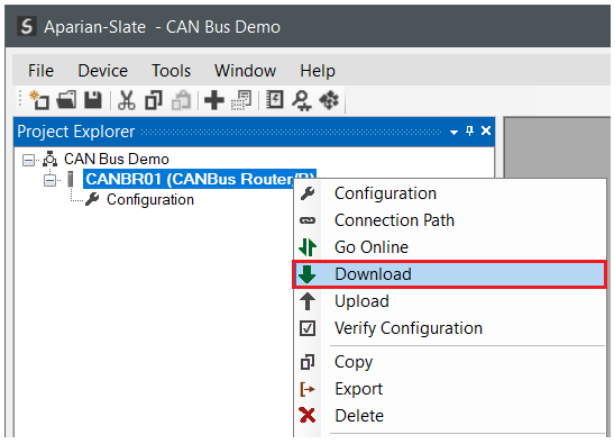


Figure 3.88 - Selecting Download

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

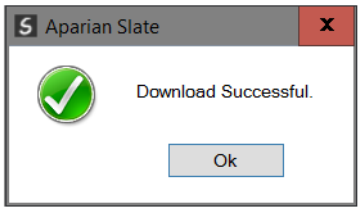


Figure 3.89 - Successful download

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.

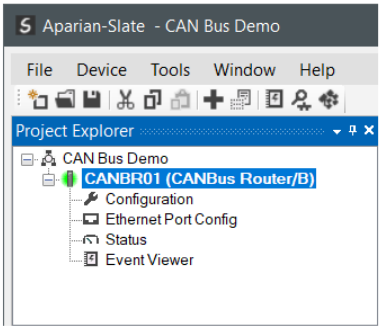


Figure 3.90 - Module online

4. SD CARD

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

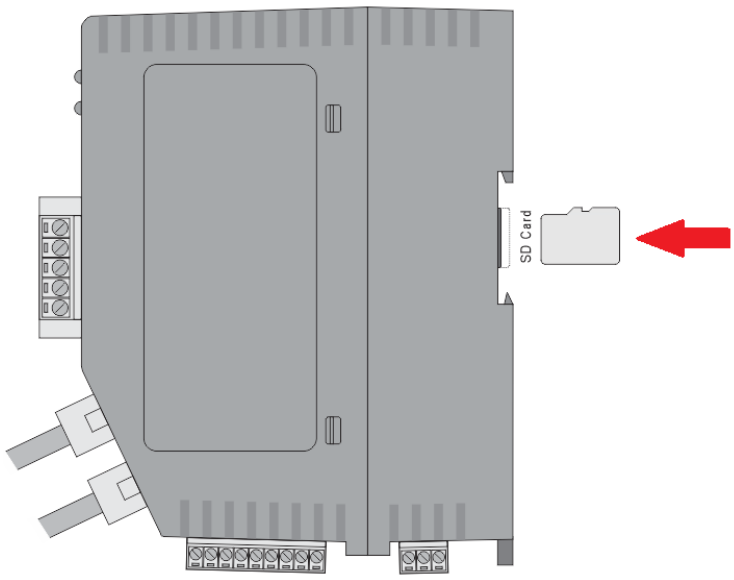


Figure 4.1 – Module Side View – SD Card Slot

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

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

4.1. FIRMWARE

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

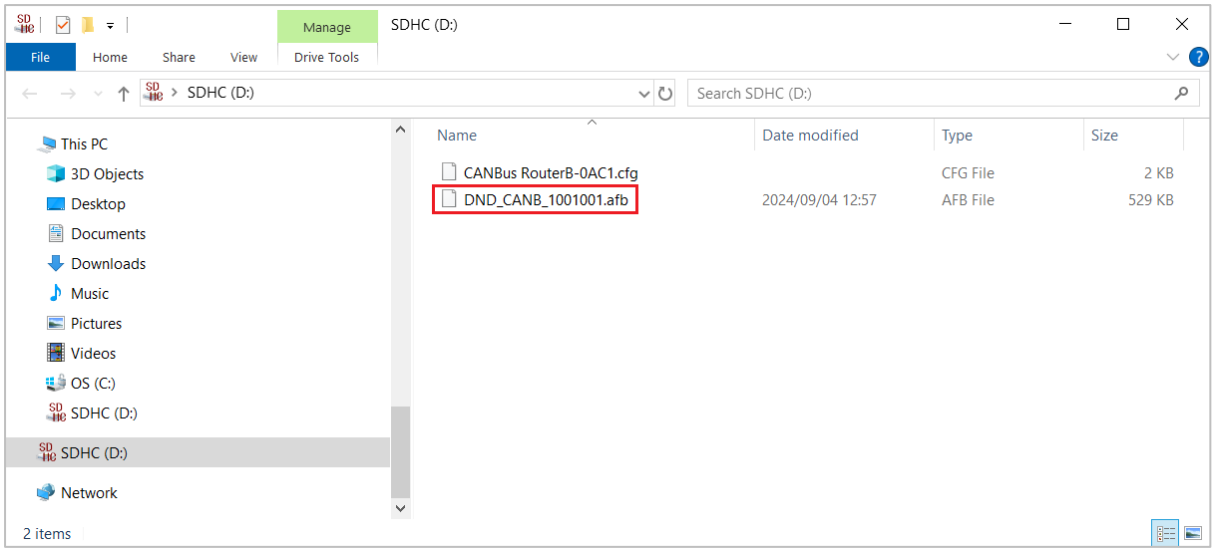


Figure 4.2 – SD Card – Firmware file

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

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

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

4.2. NETWORK PARAMETERS

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

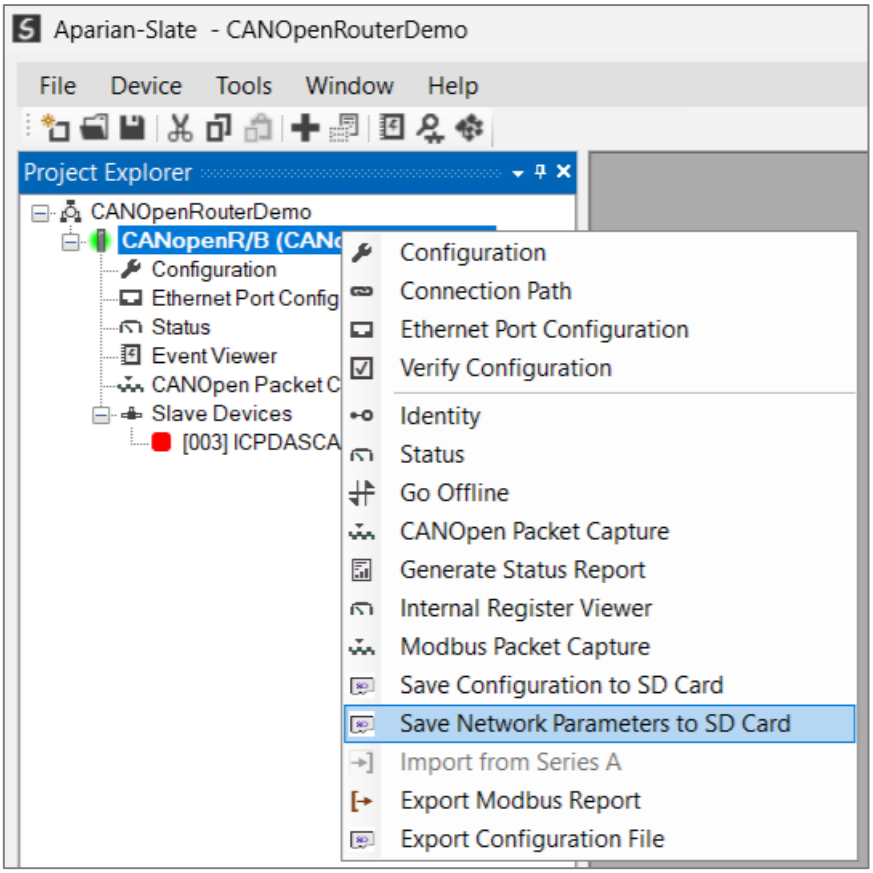


Figure 4.3 – SD Card – Network Parameters

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

4.3. CONFIGURATION

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

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

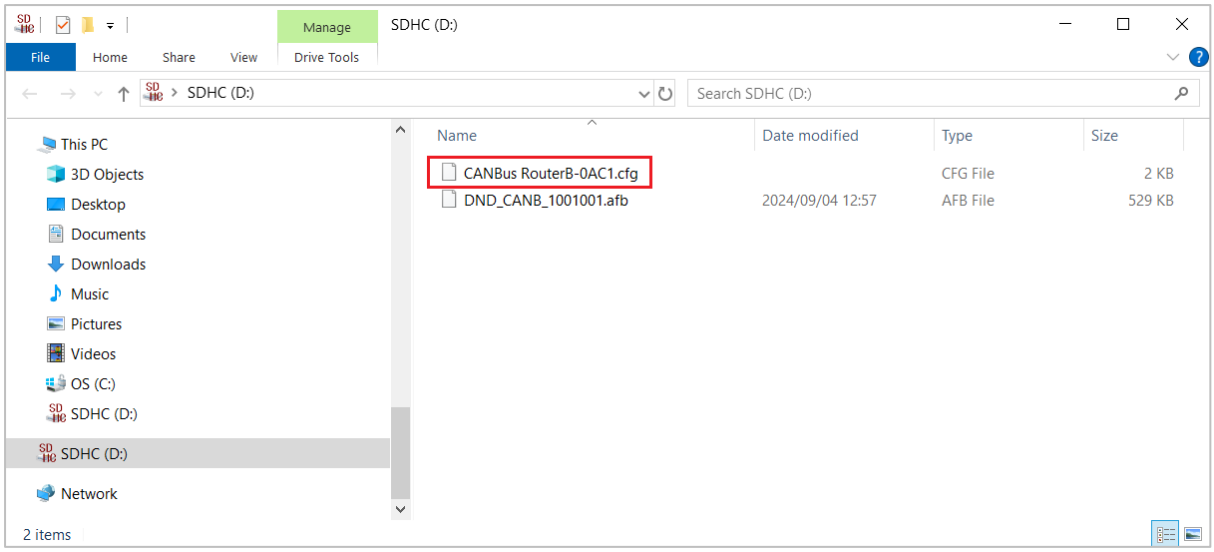


Figure 4.4 – SD Card – Configuration file

4.3.1. Manual Copy

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

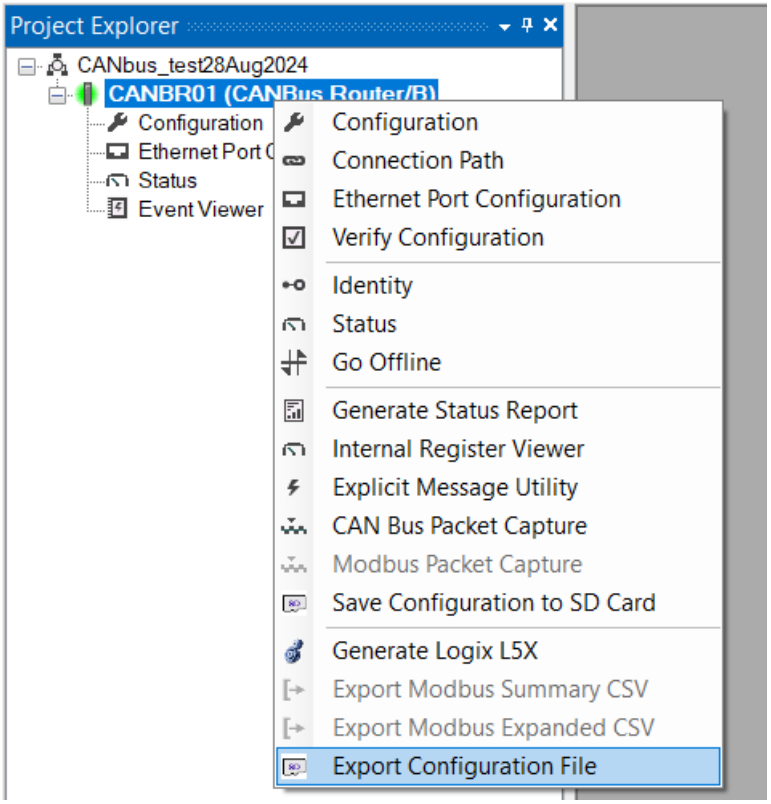


Figure 4.5 – Configuration Export for SD Card

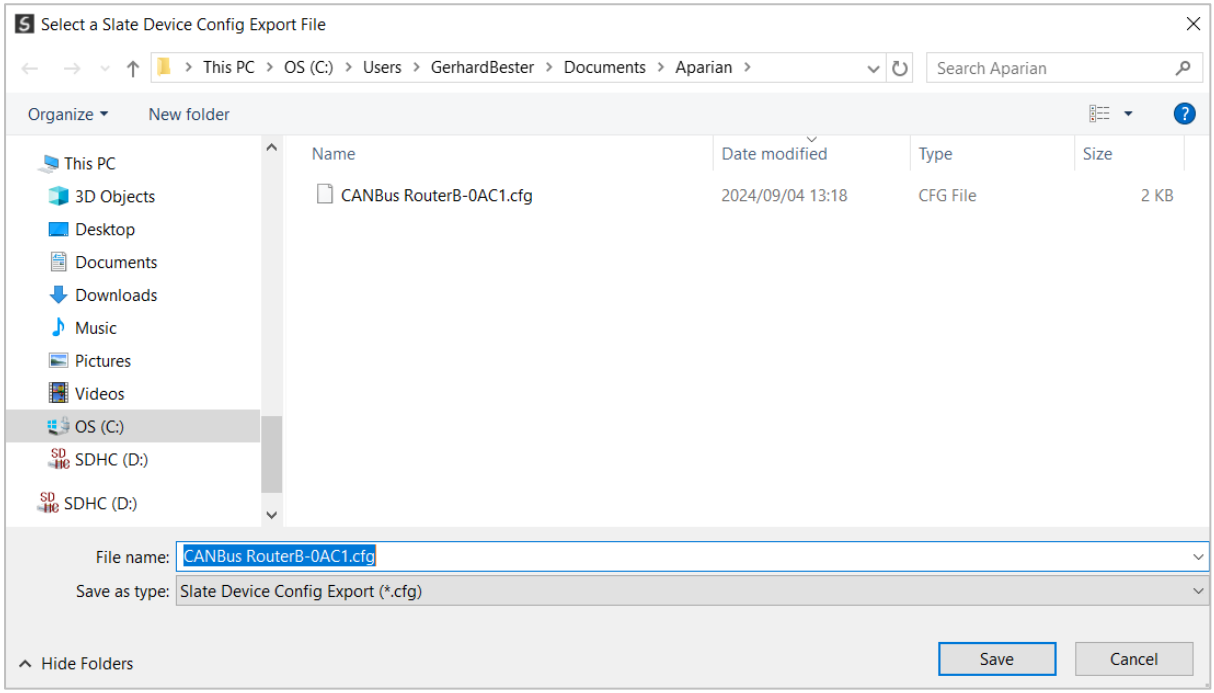


Figure 4.6 – Configuration Export for SD Card

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

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

4.3.2. SLATE TRIGGERED UPLOAD

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

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

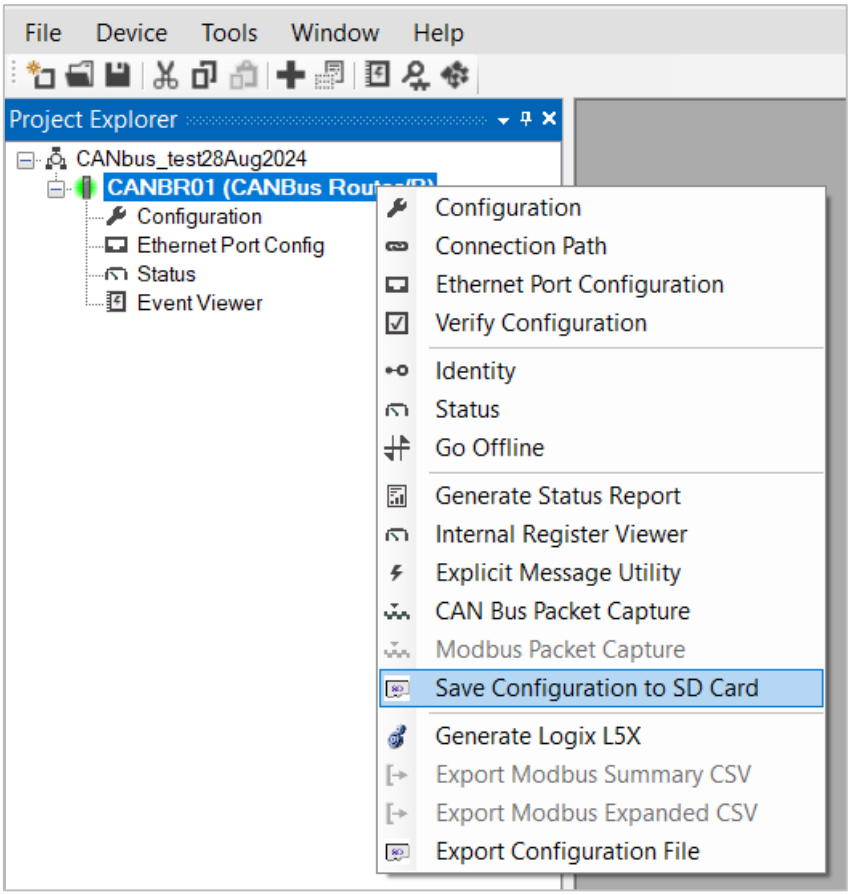


Figure 4.7 – Save Configuration to SD Card

5. DEVICE FIRMWARE UPDATE

The CAN Bus Router module supports in-field firmware upgrading. The latest firmware for the module can be downloaded from the Aparian website ***www.aparian.com***. The firmware is digitally signed, so only approved firmware can be used.

To firmware upgrade the module, follow the steps below:

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

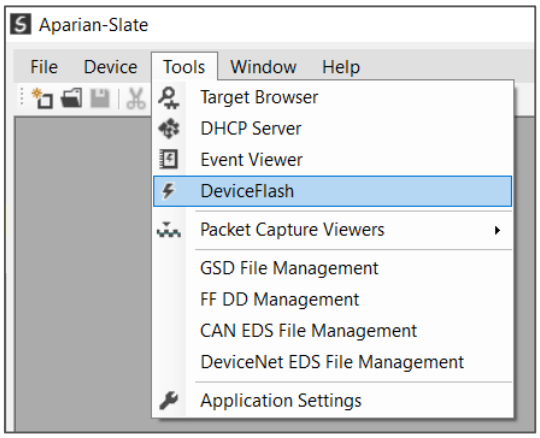


Figure 5.1 – Select DeviceFlash utility from Slate

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

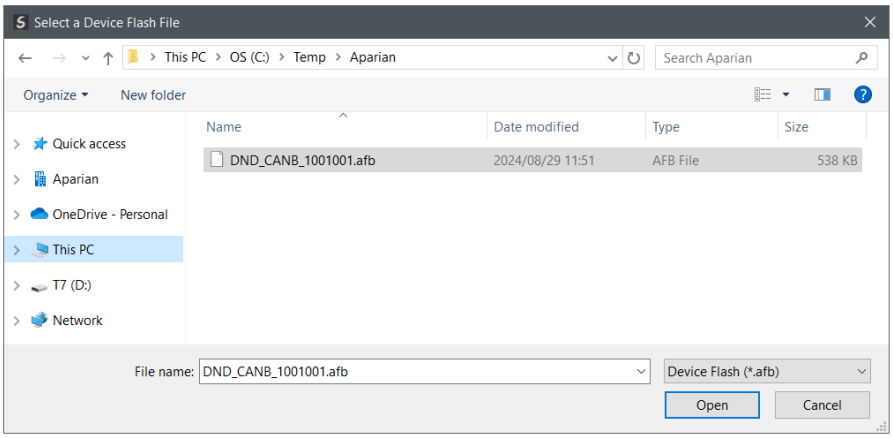


Figure 5.2 – Select the binary file

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

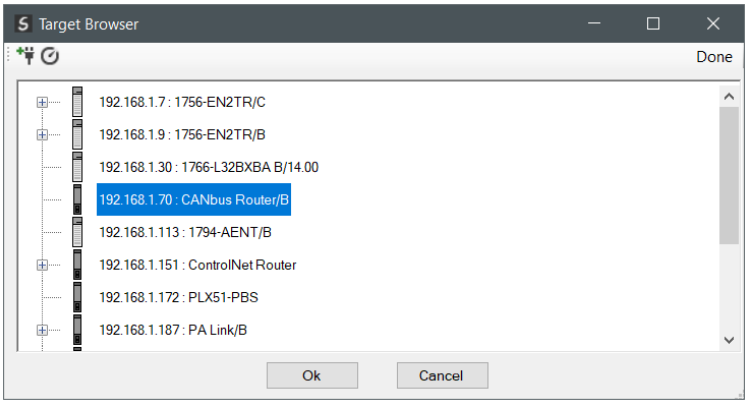


Figure 5.3 – Select the device to be updated

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

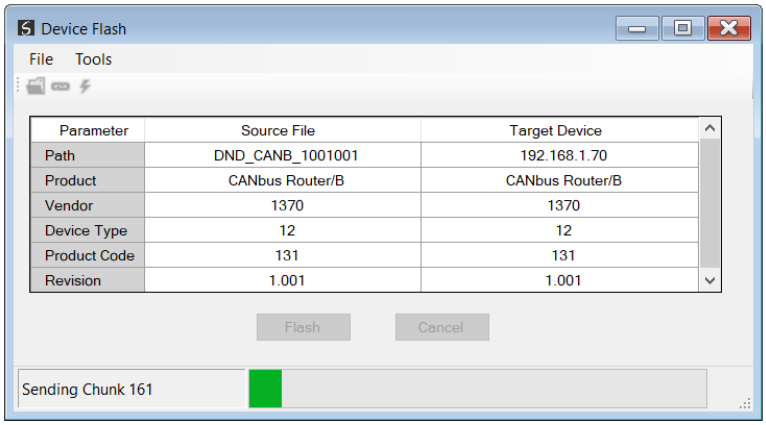


Figure 5.4 – Firmware Update Busy

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

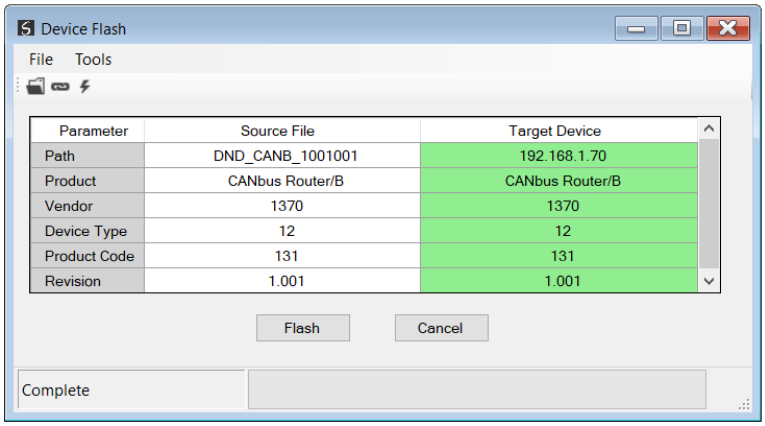


Figure 5.5 – Firmware update successfully completed.



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

6. OPERATION

6.1. CAN BUS

The CAN Bus Router can receive and transmit CAN packets on the CAN Bus network.

6.1.1. CAN RECEIVE

When a CAN packet is received by the CAN Bus Router, it is compared to each of the configured CAN Receive items, until a match is found.

A match will require the received CAN ID to match the item's ***ID Criterion*** after being bitwise-AND'ed with the configured ***ID Mask***. (See the ***CAN Receive Configuration*** section for more details.)

If the Data Match option has been selected, then the receive CAN data payload must also match the item's configured Data Criterion (after being bitwise-AND'ed with the configured ***Data Mask).***

If a match is found, then the item's Count will be incremented, and the received CAN payload data copied to the configured Internal Data Space offset.

Once a match is found for a particular CAN packet, then no further CAN Receive items will be tested.

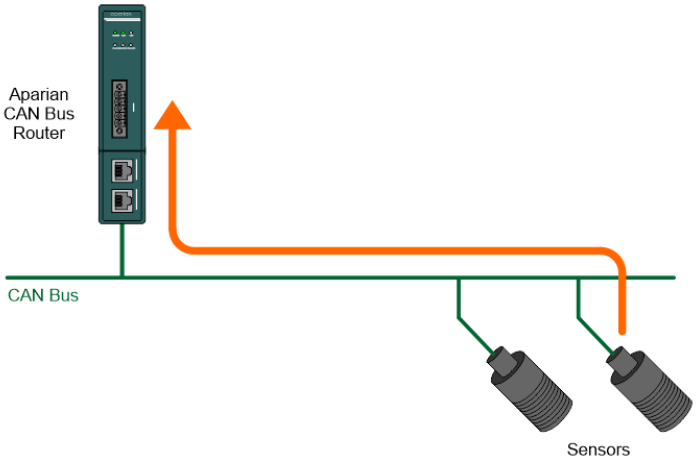


Figure 6.20 – CAN Receive data flow

6.1.2. CAN SEND

CAN packets are transmitted by the CAN Bus Router based on the CAN Send configuration. Each item (with configured CAN ID and Data) will be sent either:

- On start-up (StartUp enabled),
- At the configured interval (Periodic enabled),
- On a change of the trigger value (Trigger enabled),
- Or any combination of the above.

If the ***StartUp*** option is enabled, then the CAN packet will be sent when the module powers-up, or following the download of new module configuration.

If the ***Periodic*** option is enabled, then the CAN packet will be sent at the configured ***Interval*** (ms).

If the ***Trigger*** option is enabled, then the CAN Packet will be sent each time the 32-bit trigger value changes. The trigger value is the specified by the item's ***Trigger Offset*** in the Internal Data Space.

Note that if the CAN Packet is sent due to the trigger value changing, then the Periodic timer will be reset, and the next packet will only be sent after a complete periodic interval.

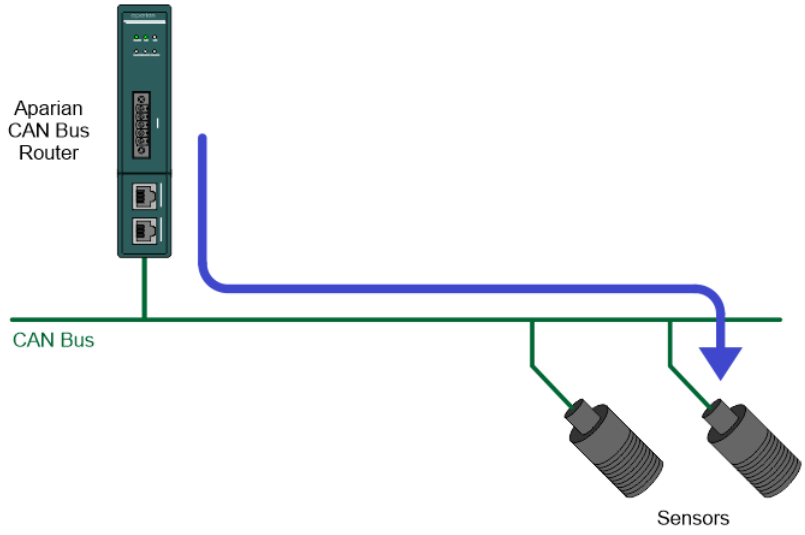


Figure 6.20 – CAN Send data flow

6.1.3. EXPLICIT CAN MESSAGING

The CAN Bus Router supports the sending of explicit CAN messages from the *Explicit Message Utility* in Slate. This is launched by right-clicking on the module, when online in Slate, and selecting ***Explicit Message Utility***.

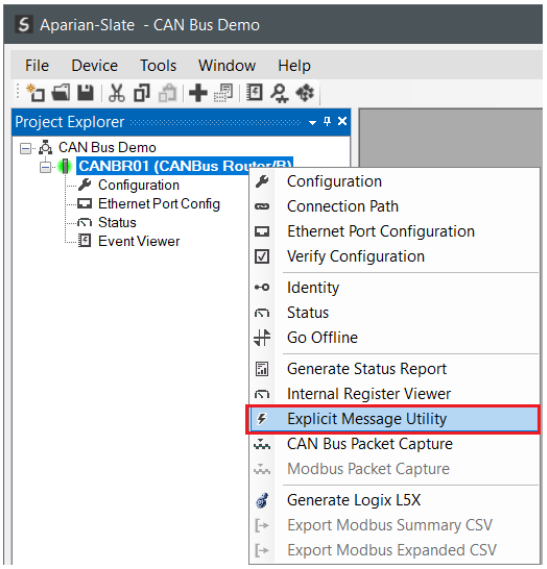


Figure 6.1 – Launching CAN Bus Explicit Messaging Utility

Once the utility is launched the user will be able to Send an ad-hoc CAN message, or CAN transaction.

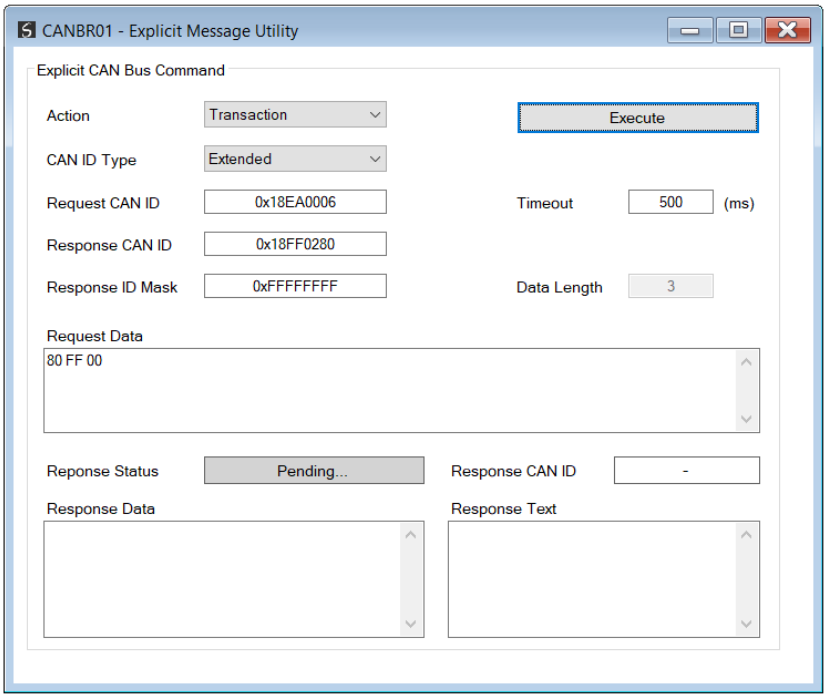


Figure 6.2 – CAN Bus Explicit Messaging Utility

Parameter	Description
Action	**Send** The configured CAN ID and payload data is transmitted on the CAN bus network. **Transaction** The configured CAN ID and payload data is transmitted on the CAN bus network. The module will then wait for a reply on the network that matches the Response CAN ID masked by the Response ID Mask and return the data.
CAN ID Type	CAN ID Type **Standard** - CAN Identifier (11-bit) **Extended** - CAN Identifier (29-bit)
Request CAN ID	The transmitted CAN ID. **NOTE**: The value can be entered as a decimal or hexadecimal. When using hexadecimal, in must start with **0x** (for example 0xA5).
Response CAN ID	The CAN ID required in a received message to be matched with this transaction. The received CAN ID, and ***Response CAN ID*** are both masked (bitwise-AND) with the ***Response ID Mask***, before making the match comparison. This allows a range of CAN IDs to be accepted as a valid response. Applicable only when the selected ***Action*** is ***Transaction.***
Response ID Mask	The CAN ID Mask to be used (bitwise-AND) when matching a received message. Each bit in the mask corresponds to the same bit location in the response, where: Mask Bit = 0: the bit in received message CAN-ID is ignored. Mask Bit = 1: the bit in the received message CAN-ID must equal the corresponding bit in the ***Response CAN ID***. For example, if: ***Response CAN ID*** = 0x354 ***Response ID Mask*** = 0xFFC Then, any message with a CAN ID of either: 0x354, 0x355, 0x356 or 0x357 will be a match. Applicable only when the selected ***Action*** is ***Transaction.***
Timeout	The number of milliseconds before the message times out.
Data Length	The number of bytes to send. **NOTE:** This is read-only and will automatically update as data is entered into the Request Data window.
Request Data	This is the data that must be sent with the CAN message. **NOTE:** The data must be entered in hexadecimal and delimited by spaces (as shown below): 10 AB E5 33

Table 6.1 - CAN Bus Explicit Messaging Utility parameters

Once the relevant parameters and data have been setup and populated, then the user can press the *Execute* button to send the CAN request onto the CAN Bus network via the CAN Bus Router.

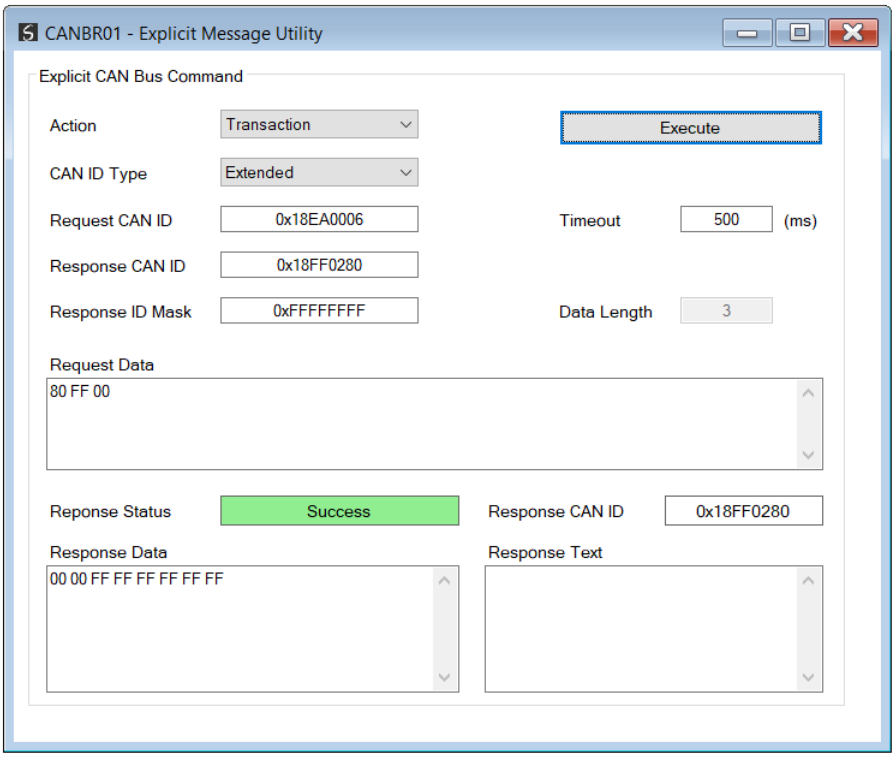


Figure 6.3 – CAN Bus Explicit Messaging Utility Executed

6.2. ETHERNET/IP TARGET

A controller (e.g. Logix controller) can own the CAN Bus Router over EtherNet/IP using up to 4 Class 1 EtherNet/IP connections when the CAN Bus Router is operating as an EtherNet/IP target. This will allow the CAN Bus Router to exchange data with the controller using the input and output assembly of the Class 1 EtherNet/IP connection.



NOTE: When using EtherNet/IP Target, it is recommended to use the ***Recommend*** button in the Internal Map configuration. This will automatically map and reformat all the required data in the Internal Map.

6.2.1. CLASS 1 ASSEMBLY MAPPING

When the module operates in a Logix “owned” mode the Logix controller will establish a class 1 cyclic communication connection to the CAN Bus Router. Up to four input and output assemblies are exchanged at a fix interval (RPI).

NOTE: The module input and output assembly of each connection will be an undecorated array of bytes. The imported Logix routine (generated by Slate) will copy this data to the input and output assemblies.

Once the generate L5X file has been imported (which will match the Internal Mapping in the configuration), the user will be able to use the tags generated for the specific CAN Bus Router.

The CAN Bus Router System tag is shown below:

Name	Style	Data Type	Description
CANB01System		CANBusSystemStatus	
CANB01System.ConfigValid	Decimal	BOOL	Configuration Valid
CANB01System.EIPOriginatorCommsOk	Decimal	BOOL	EtherNet/IP Originator 0=Fail, 1=Ok
CANB01System.ModbusOnline	Decimal	BOOL	0=Offline, 1=Online
CANB01System.EIPOwned	Decimal	BOOL	EtherNet/IP Target: 0=Not-Owned, 1=Owned
CANB01System.PowerMainConnector	Decimal	BOOL	Main Power: 0=Off 1=On
CANB01System.PowerCANConnector	Decimal	BOOL	CAN Power: 0=Off 1=On
CANB01System.ControllerRun	Decimal	BOOL	Controller Mode: 0=Program, 1=Run
CANB01System.NTPOk	Decimal	BOOL	NTP Status: 0=Fail, 1=Ok
CANB01System.CANBusOk	Decimal	BOOL	CAN Bus Status: 0=Fail, 1=Ok
CANB01System.ConfigCRC	Hex	INT	Configuration Checksum
CANB01System.TransactionRate	Decimal	INT	Transactions per second
CANB01System.CANBaud	Decimal	SINT	Current CAN Baud: 0=10k, 1=20k, 2=50k, 3=125k, 4=250k, 5=500k, 6=800k, 7...
CANB01System.Temperature	Float	REAL	Module Temperature (deg C)
CANB01System.CANRxPacketCount	Decimal	DINT	CAN Rx Packet Count
CANB01System.CANTxPacketCount	Decimal	DINT	CAN Tx Packet Count
CANB01System.CANCRCErrors	Decimal	DINT	CAN CRC Errors
CANB01System.CANBitErrors	Decimal	DINT	CAN Bit Errors
CANB01System.CANStuffErrors	Decimal	DINT	CAN Stuff Errors
CANB01System.CANBusOffCount	Decimal	DINT	CAN Bus-Off Events
CANB01System.CANAckErrors	Decimal	DINT	CAN Acknowledgment Errors

Figure 6.4 – Logix System Status Tag

An example of a CAN Bus Receive item's tags are shown below:

Name	Value	Style	Data Type
CANBR01DashKeys_Count	2134	Decimal	DINT
CANBR01DashKeys_Data	{...}	Decimal	SINT[8]

Figure 6.5 – CAN Receive specific tags

An example of a CAN Bus Send item's tags are shown below:

Name	Value	Style	Data Type
CANBR01DashLED_Count	423	Decimal	DINT
CANBR01DashLED_Data	{...}	Decimal	SINT[8]
CANBR01DashLED_Trigger	10	Decimal	DINT

Figure 6.6 – CAN Send specific tags

6.2.2. EXPLICIT MESSAGING

The CAN Bus Router allows the user to read PGN data from a CAN Bus IO devices using explicit EtherNet/IP CIP messages. The required parameters for CAN Bus PGN extraction from a CAN Bus IO device are listed below.

6.2.2.1. EXPLICIT CAN MESSAGE

A. CIP MESSAGE

Parameter	Description
Service Code	0x6D (Hex)
Class	0x441 (Hex)
Instance	1
Attribute	N/A
Request Data Length	28

Table 6.2 – Explicit CAN Message

NOTE: The Explicit CAN Message Request and Response UDTs are included in the L5X file, when selecting the ***Generate Logix L5X*** option.

B. REQUEST DATA

Parameter	Data Type	Description
Command	SINT	**0 – Send** The configured CAN ID and payload data is transmitted on the CAN bus network. **1 – Transaction** The configured CAN ID and payload data is transmitted on the CAN bus network. The module will then wait for a reply on the network that matches the ***Response CAN ID*** masked by the ***Response ID Mask*** and return the data.
CAN ID Type	SINT	CAN ID Type 0 – ***Standard*** CAN Identifier (11-bit) 1 – ***Extended*** CAN Identifier (29-bit)
Reserved	INT	*(Reserved)*

Request CAN ID	DINT	The transmitted CAN ID.
Response CAN ID	DINT	The CAN ID required in a received message to be matched with this transaction. The received CAN ID, and Response CAN ID are both masked (bitwise-AND) with the Response ID Mask, before making the match comparison. This allows a range of CAN IDs to be accepted as a valid response. Applicable only when the selected ***Command*** is ***Transaction***.
Response ID Mask	DINT	The CAN ID Mask to be used (bitwise-AND) when matching a received message. Each bit in the mask corresponds to the same bit location in the response, where: Mask Bit = 0: the bit in received message CAN-ID is ignored. Mask Bit = 1: the bit in the received message CAN-ID must equal the corresponding bit in the ***Response CAN ID***. For example, if: ***Response CAN ID*** = 0x354 ***Response ID Mask*** = 0xFFC Then, any message with a CAN ID of either: 0x354, 0x355, 0x356 or 0x357 will be a match. Applicable only when the selected ***Command*** is ***Transaction***.
Timeout	INT	The time (in milliseconds) before the message transaction is marked as failed in. Default 500ms. Applicable only when the selected ***Action*** is ***Transaction***.
Data Length	INT	The size of the data to be sent (0-8 bytes).
Data	SINT[]	The data to be sent.

Table 6.3 – Explicit CAN Message Request

C. RESPONSE DATA

Parameter	Data Type	Description
Response Status	INT	This is the status of the request. 0 – Success 1 – Timeout
Data Length	INT	The size of the response data to follow (0-8 bytes). Applicable only when the ***Command*** is **Transaction.**
Response ID	DINT	The CAN ID of the message that was matched as a suitable reply. Applicable only when the ***Command*** is **Transaction.**
Data	SINT[]	The response data from the target CAN Bus device.

		Applicable only when the ***Command*** is **Transaction.**

Table 6.4 – Explicit CAN Message Response

6.2.2.2. SET CAN BAUD RATE

A. CIP MESSAGE

Parameter	Description
Service Code	0x10 (Hex) – Set Single Attribute
Class	0x441 (Hex)
Instance	1
Attribute	16
Request Data Length	1

Table 6.5 – Set CAN Baud Rate Message

B. REQUEST DATA

Parameter	Data Type	Description
Baud Rate Option	SINT	0 – 10k 1 – 20k 2 – 50k 3 – 125k 4 – 250k 5 – 500k 6 – 800k 7 – 1M

Table 6.6 – Set CAN Baud Rate Request

C. RESPONSE DATA

No data is returned for a Set Single Attribute message.

6.3. ETHERNET/IP ORIGINATOR

The CAN Bus Router module can operate as an EtherNet/IP originator. In this mode the module can exchange data from the CAN Bus network with EtherNet/IP devices using either the input and output assemblies of the Class 1 EtherNet/IP connection to the device or using explicit (Class 3 or UCMM) EtherNet/IP messages.

6.3.1. ETHERNET/IP CLASS 1 CONNECTIONS

In the example below, the CAN Bus Router is sending, and receiving, CAN packets while also owning two EtherNet/IP IO modules. The CAN data is being exchanged with the EtherNet/IP IO modules.

Firstly, the CAN Receive tab is configured to receive the relevant CAN packets. In this example the CAN bus keypad is transmitting the keypad info on CAN ID 0x18FF0280 and this data is being mapped to the Internal Data Space at offset 4000. Secondly, the first 2 bytes of this data (Internal Data Space) is mapped to the Flex output module (1794-OA16).

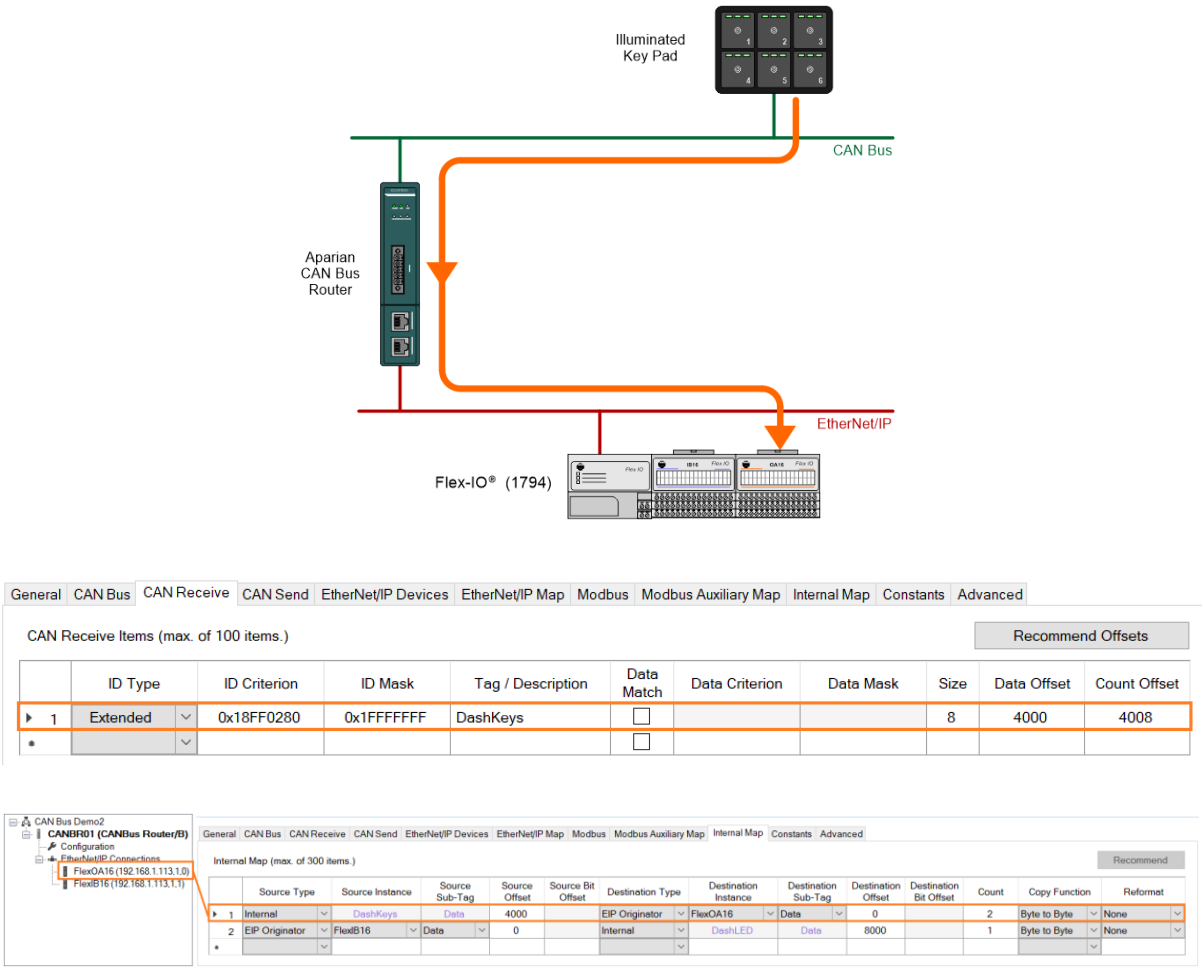


Figure 6.7 – Internal Mapping from CAN Bus to EtherNet/IP Originator

Similarly, but in the reverse direction, the data received from the Flex input module (1794-IB16) is mapped to the Internal Data Space at offset 8000.

Using the CAN Send tab, this data (Internal Data Space at offset 8000) is sent to Key Pad to control its LEDs.

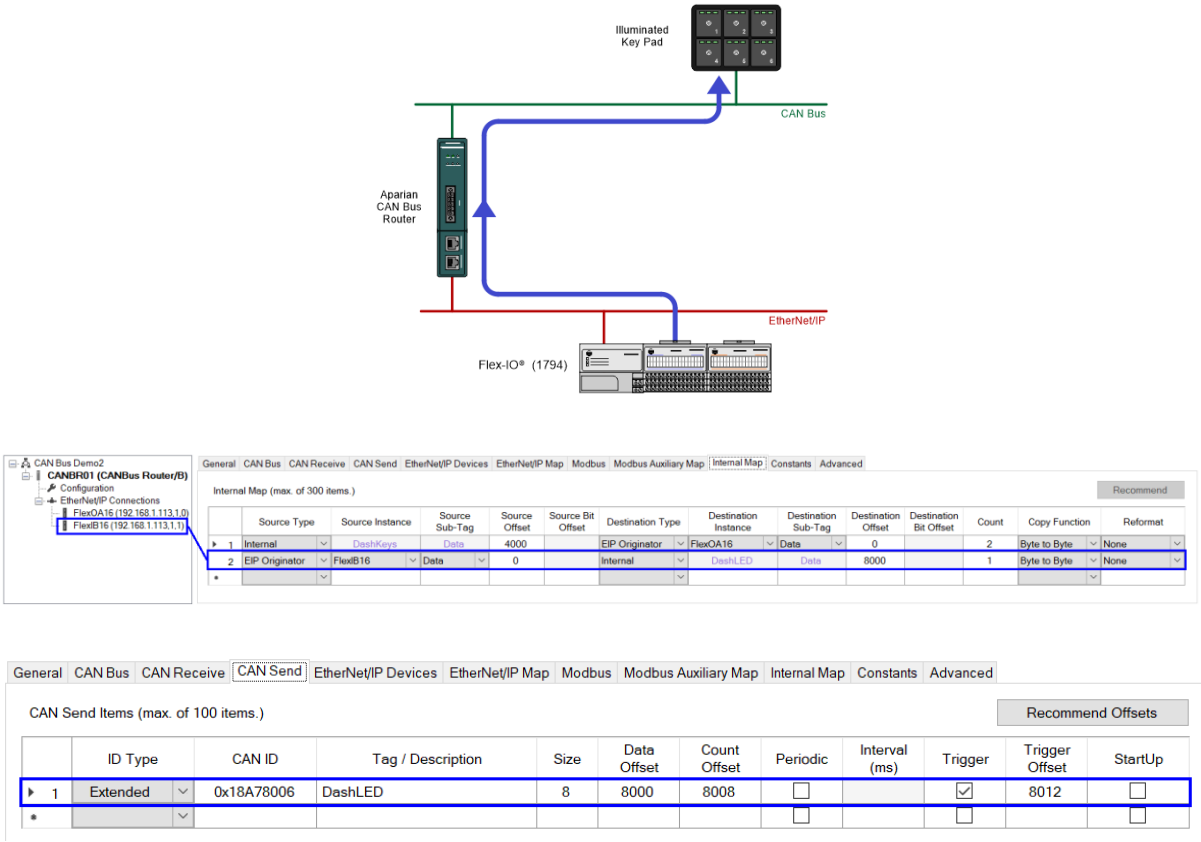


Figure 6.8 – Internal Mapping from EtherNet/IP Originator to CAN Bus

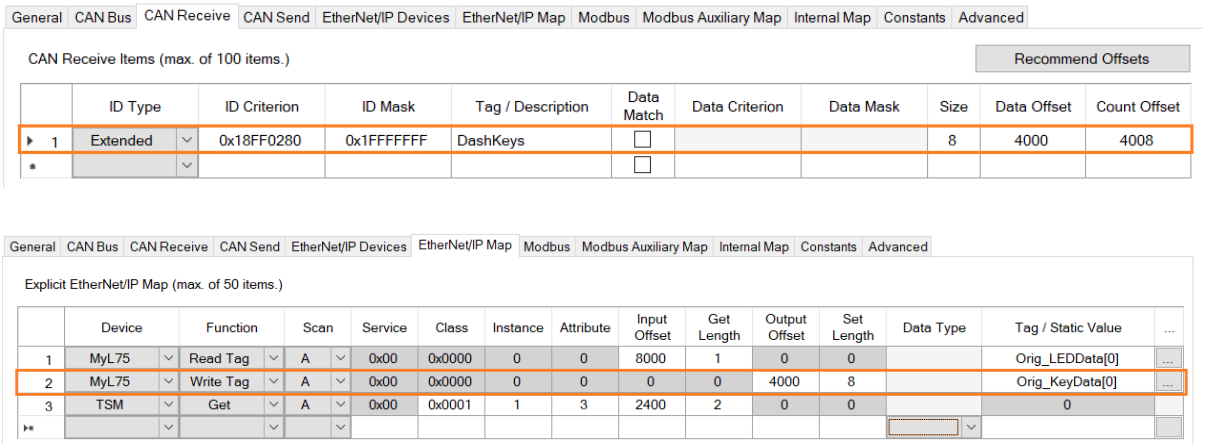
6.3.2. EXPLICIT ETHERNET/IP MESSAGING

When using the EtherNet/IP Explicit Messaging, the user can configure up to 10 EtherNet/IP devices which will be used for the Explicit Messaging. This configuration is accessed in the *EtherNet/IP Devices* tab. Following this, the EtherNet/IP Map of explicit messages needs to be configured. The Explicit Messaging uses the internal data space (IDS) where data can be stored for exchanges between the explicit EtherNet/IP devices and the CAN Bus network.

The Input and Output IDS Offset is where the Explicit EtherNet/IP device data will be read from, or written to. The data in the IDS can then, in turn, be copied to or from the CAN Bus network using the CAN Receive and CAN Send configuration.

In the below example, the CAN bus keypad is transmitting the keypad info on CAN ID 0x18FF0280 and this data is being mapped to the Internal Data Space at offset 4000.

Secondly, in the Explicit EtherNet/IP Map, the same data (IDS offset 4000) is written to the Logix tag (SINT Array) “Orig_KeyData[0]”.



General | CAN Bus | CAN Receive | CAN Send | EtherNet/IP Devices | EtherNet/IP Map | Modbus | Modbus Auxiliary Map | Internal Map | Constants | Advanced

CAN Receive Items (max. of 100 items.) — Recommend Offsets

	ID Type	ID Criterion	ID Mask	Tag / Description	Data Match	Data Criterion	Data Mask	Size	Data Offset	Count Offset
1	Extended	0x18FF0280	0x1FFFFFFF	DashKeys	☐			8	4000	4008
*					☐					

General | CAN Bus | CAN Receive | CAN Send | EtherNet/IP Devices | EtherNet/IP Map | Modbus | Modbus Auxiliary Map | Internal Map | Constants | Advanced

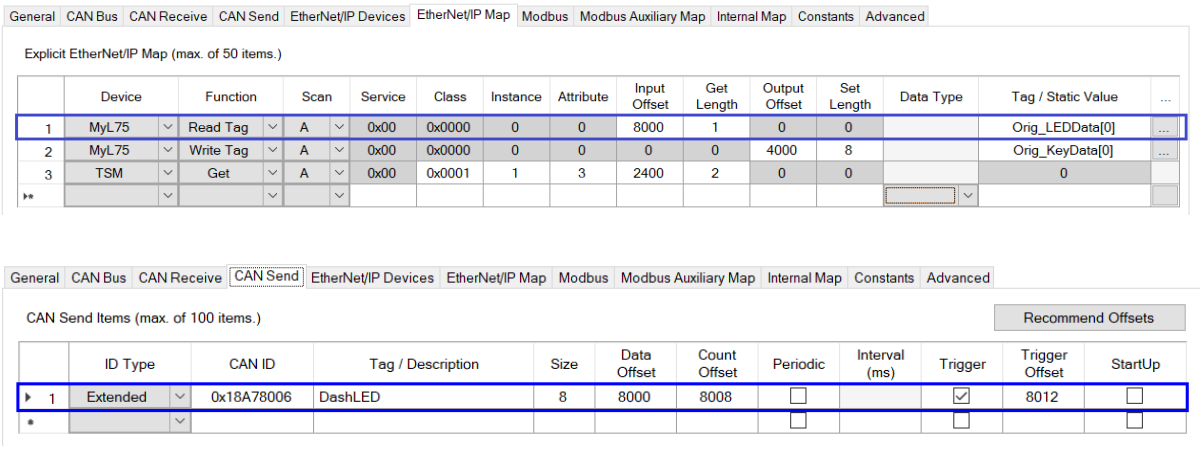
Explicit EtherNet/IP Map (max. of 50 items.)

	Device	Function	Scan	Service	Class	Instance	Attribute	Input Offset	Get Length	Output Offset	Set Length	Data Type	Tag / Static Value	...
1	MyL75	Read Tag	A	0x00	0x0000	0	0	8000	1	0	0		Orig_LEDData[0]	...
2	MyL75	Write Tag	A	0x00	0x0000	0	0	0	0	4000	8		Orig_KeyData[0]	...
3	TSM	Get	A	0x00	0x0001	1	3	2400	2	0	0		0	
**														

Figure 6.9 – Received CAN Bus Data to Explicit Logix Tag

In the Explicit EtherNet/IP Map, the Logix tag “Orig_LEDData[0]” (SINT Array) is read from the Logix controller “MyL75” and the data is copied to Internal Data Space at offset 8000.

This data is then transmitted to CAN Bus keypad as specified in the CAN Send configuration.



General | CAN Bus | CAN Receive | CAN Send | EtherNet/IP Devices | EtherNet/IP Map | Modbus | Modbus Auxiliary Map | Internal Map | Constants | Advanced

Explicit EtherNet/IP Map (max. of 50 items.)

	Device	Function	Scan	Service	Class	Instance	Attribute	Input Offset	Get Length	Output Offset	Set Length	Data Type	Tag / Static Value	...
1	MyL75	Read Tag	A	0x00	0x0000	0	0	8000	1	0	0		Orig_LEDData[0]	...
2	MyL75	Write Tag	A	0x00	0x0000	0	0	0	0	4000	8		Orig_KeyData[0]	...
3	TSM	Get	A	0x00	0x0001	1	3	2400	2	0	0		0	
**														

General | CAN Bus | CAN Receive | CAN Send | EtherNet/IP Devices | EtherNet/IP Map | Modbus | Modbus Auxiliary Map | Internal Map | Constants | Advanced

CAN Send Items (max. of 100 items.) — Recommend Offsets

	ID Type	CAN ID	Tag / Description	Size	Data Offset	Count Offset	Periodic	Interval (ms)	Trigger	Trigger Offset	StartUp
1	Extended	0x18A78006	DashLED	8	8000	8008	☐		☑	8012	☐
*							☐		☐		☐

Figure 6.10 – Explicit Logix Tag Read to CAN Bus

6.4. MODBUS CLIENT

When the CAN Bus Router has the Primary Interface set to Modbus Client, then the CAN Bus data can be exchanged with one or more remote Modbus servers.

The internal Modbus Registers are asynchronously exchanged with Modbus devices as configured in the Modbus Auxiliary Map. In this mapping the user can exchange (read or write) data between the internal Modbus Registers and a remote Modbus device on Modbus TCP, RTU232, or RTU485.

In the example below the CAN Bus Router with the Primary Interface set to Modbus Client will read a Modbus Holding Register from a Modbus Server device and then transmit this data to the CAN Bus keypad.

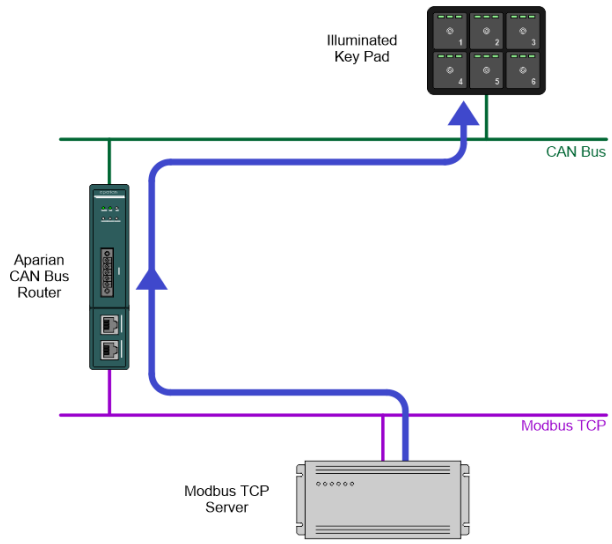
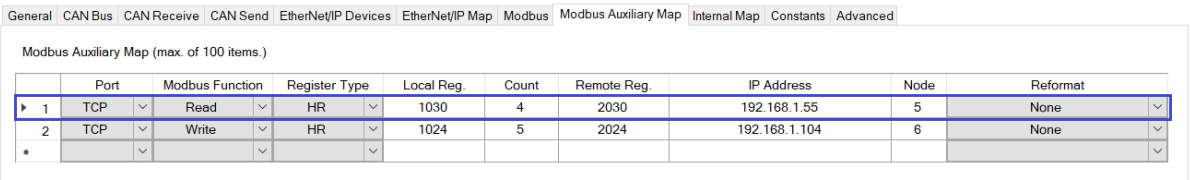


Figure 6.11 – Modbus Client to CAN Bus operation

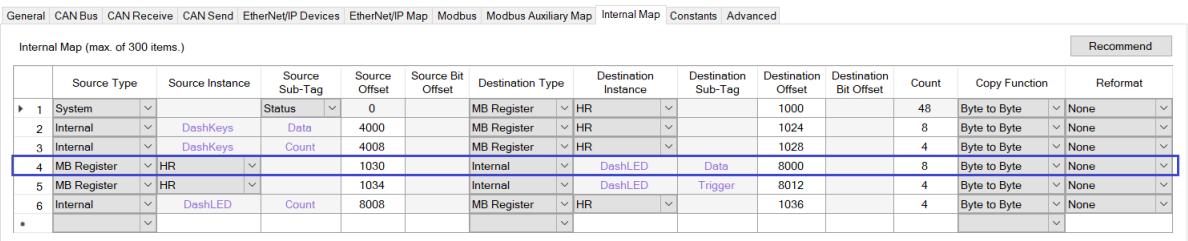
Firstly, the Modbus Auxiliary Map is configured to read Holding Register 2030-2033 from the remote Modbus server at IP address 192.168.1.55 and store the result in the internal Modbus Holding Register 1030-1033.



	Port	Modbus Function	Register Type	Local Reg.	Count	Remote Reg.	IP Address	Node	Reformat
1	TCP	Read	HR	1030	4	2030	192.168.1.55	5	None
2	TCP	Write	HR	1024	5	2024	192.168.1.104	6	None
*									

Figure 6.12 – Modbus Client Auxiliary Mapping

In the Internal Map configuration, the data in Modbus Holding registers 1030-1033 is transferred to the Internal Data Space starting at offset 8000.



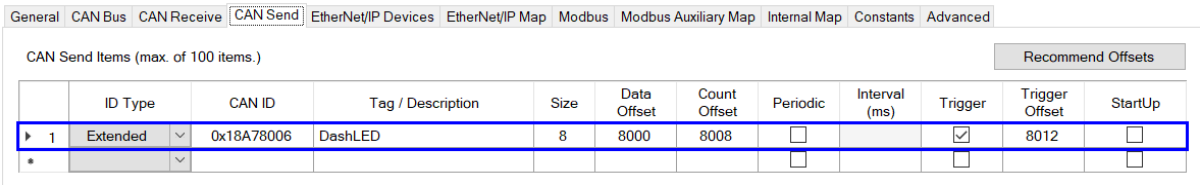
General | CAN Bus | CAN Receive | CAN Send | EtherNet/IP Devices | EtherNet/IP Map | Modbus | Modbus Auxiliary Map | Internal Map | Constants | Advanced

Internal Map (max. of 300 items.) — Recommend

	Source Type	Source Instance	Source Sub-Tag	Source Offset	Source Bit Offset	Destination Type	Destination Instance	Destination Sub-Tag	Destination Offset	Destination Bit Offset	Count	Copy Function	Reformat
1	System		Status	0		MB Register	HR		1000		48	Byte to Byte	None
2	Internal	DashKeys	Data	4000		MB Register	HR		1024		8	Byte to Byte	None
3	Internal	DashKeys	Count	4008		MB Register	HR		1028		4	Byte to Byte	None
4	MB Register	HR		1030		Internal	DashLED	Data	8000		8	Byte to Byte	None
5	MB Register	HR		1034		Internal	DashLED	Trigger	8012		4	Byte to Byte	None
6	Internal	DashLED	Count	8008		MB Register	HR		1036		4	Byte to Byte	None

Figure 6.13 – Modbus Client - Internal Mapping

Finaly, in the CAN Send configuration, the data in the Internal Data Space at offset 8000 is transmitted to the CAN Bus key pad.



General | CAN Bus | CAN Receive | CAN Send | EtherNet/IP Devices | EtherNet/IP Map | Modbus | Modbus Auxiliary Map | Internal Map | Constants | Advanced

CAN Send Items (max. of 100 items.) — Recommend Offsets

	ID Type	CAN ID	Tag / Description	Size	Data Offset	Count Offset	Periodic	Interval (ms)	Trigger	Trigger Offset	StartUp
1	Extended	0x18A78006	DashLED	8	8000	8008	☐		☑	8012	☐

Figure 6.14 – Modbus Client – CAN Send

6.5. MODBUS SERVER

When the CAN Bus Router has the Primary Interface set to Modbus Server, then the CAN Bus data can be mapped to and from configurable internal Modbus Registers and offsets using the Internal Map.

The internal Modbus Registers can then be asynchronously exchanged with one or more remote Modbus Clients on Modbus TCP, RTU232, or RTU485.

In the example below the CAN Bus Router with the Primary Interface set to Modbus Server, receives data from the CAN Bus keypad and makes it available to remote Modbus Clients.

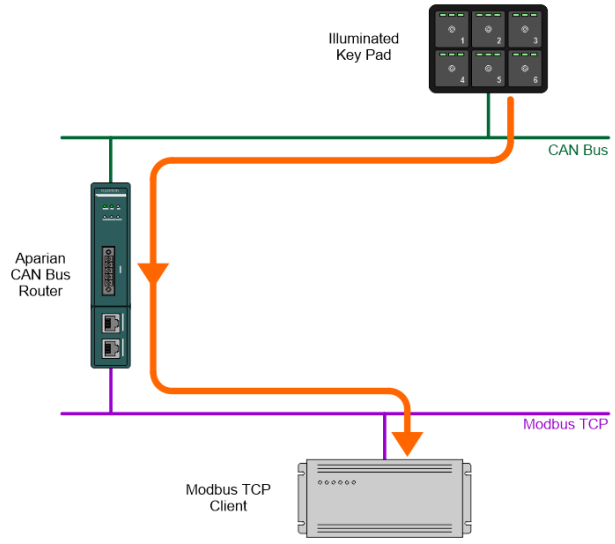


Figure 6.15 – CAN Bus to Modbus TCP Client

Firstly, the CAN Receive tab is configured to receive the keypad data (CAN ID 0x18FF0280) and copy this data to the Internal Data Space at offset 4000.

General | CAN Bus | CAN Receive | CAN Send | EtherNet/IP Devices | EtherNet/IP Map | Modbus | Modbus Auxiliary Map | Internal Map | Constants | Advanced

CAN Receive Items (max. of 100 items.) — Recommend Offsets

	ID Type	ID Criterion	ID Mask	Tag / Description	Data Match	Data Criterion	Data Mask	Size	Data Offset	Count Offset
1	Extended	0x18FF0280	0x1FFFFFFF	DashKeys	☐			8	4000	4008
*					☐					

Figure 6.16 – Modbus Server – CAN Send

In the Internal Mapping configuration, this data (Internal Data Space at offset 4000) is mapped to internal Modbus Holding Register 1024-1027.

General | CAN Bus | CAN Receive | CAN Send | EtherNet/IP Devices | EtherNet/IP Map | Modbus | Modbus Auxiliary Map | Internal Map | Constants | Advanced

Internal Map (max. of 300 items.) — Recommend

	Source Type	Source Instance	Source Sub-Tag	Source Offset	Source Bit Offset	Destination Type	Destination Instance	Destination Sub-Tag	Destination Offset	Destination Bit Offset	Count	Copy Function	Reformat
1	System		Status	0		MB Register	HR		1000		48	Byte to Byte	None
2	Internal	DashKeys	Data	4000		MB Register	HR		1024		8	Byte to Byte	None
3	Internal	DashKeys	Count	4008		MB Register	HR		1028		4	Byte to Byte	None
4	MB Register	HR		1030		Internal	DashLED	Data	8000		8	Byte to Byte	None
5	MB Register	HR		1034		Internal	DashLED	Trigger	8012		4	Byte to Byte	None
6	Internal	DashLED	Count	8008		MB Register	HR		1036		4	Byte to Byte	None
*													

Figure 6.17 – Modbus Server –Internal Mapping

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

7. DIAGNOSTICS

7.1. LEDS

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

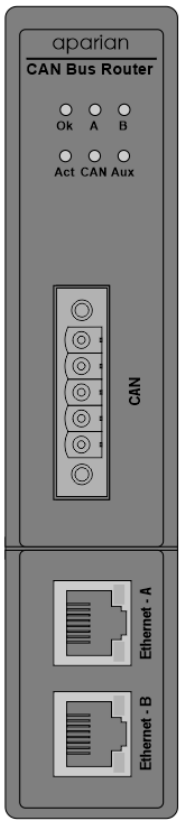


Figure 7.1 – CAN Bus Router front view

LED	Description
Ok	The module LED will provide information regarding the system-level operation of the module. If the LED is **red**, then the module is not operating correctly. For example, if the module application firmware has been corrupted or there is a hardware fault the module will have a red Module LED. If the LED is briefly **flashing red**, and then returning to either flashing green or solid green, then there is a duplicate IP address on the Ethernet network similar to that of the local module.

	If the LED is **green (flashing)**, then the module has booted and is running correctly **without** any application configuration loaded. If the LED is **green (solid)**, then the module has booted and is running correctly **with** application configuration loaded.
A / B	The Ethernet LED will light up when an Ethernet link has been detected (by plugging in a connected Ethernet cable). The LED will flash every time traffic was detected. This module has two Ethernet ports A and B. Each LEDs represents each specific port.
Act	The Act LED indicates if the module is currently in an active state. Flashing Green – The LED flashes green each time a valid CAN packet is received which matches the criteria of an item in the CAN Receive list. Off - The local CAN Bus Router is **not** sending or receiving any valid CAN packets.
CAN	The CAN LED indicates the activity on the CAN Bus network. Flashing Red – A corrupted or incorrect CAN Bus packet was received. Flashing Green – A valid CAN Bus packet was received. Off – No CAN Bus packets are being received or sent.
Aux	The Aux LED will flash each time there was activity on any of the primary interfaces. Flashing Red – A corrupted or incorrect packet was received on one of the Primary Interfaces (EtherNet/IP, Modbus TCP/RTU232/RTU485). Flashing Green – A valid packet was received on one of the Primary Interfaces (EtherNet/IP, Modbus TCP/RTU232/RTU485). Off – No activity.

Table 7.1 - Module LED operation

7.2. MODULE STATUS MONITORING IN SLATE

The CAN Bus Router provides various statistics which can assist with module operation, maintenance, and fault finding. The statistics can be accessed in full by Slate or using the web server in the module.

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

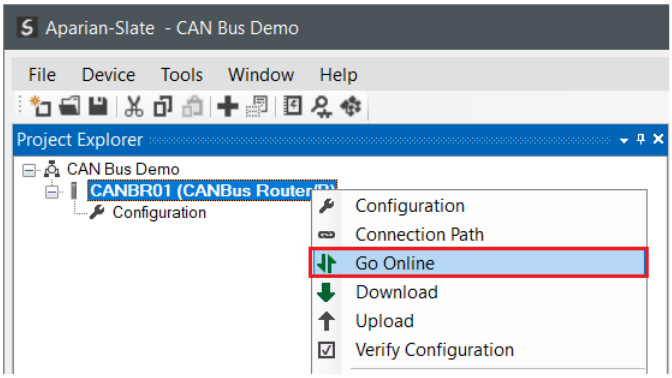


Figure 7.2 - Selecting to Go Online

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

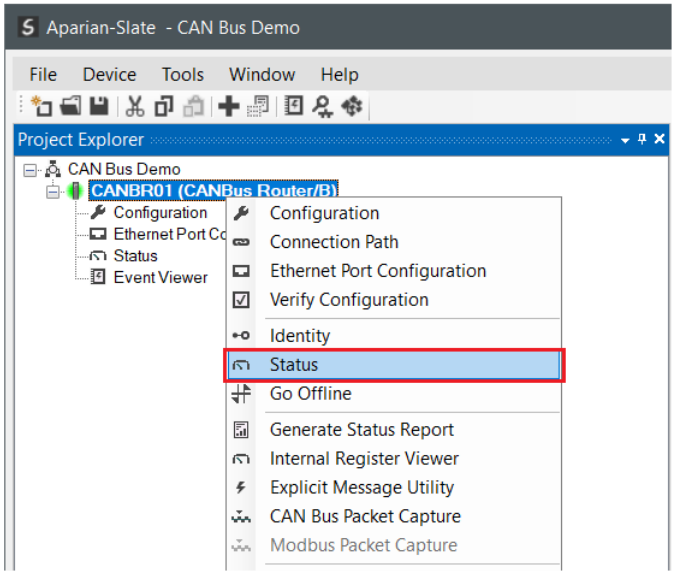


Figure 7.3 - Selecting online Status

The Status monitoring window can be opened by either double-clicking on the ***Status*** item in the Project Explorer tree, or by right-clicking on the module and selecting ***Status***. The status window contains multiple tabs to display the current status of the module.

7.2.1. GENERAL

The General tab displays the general status for the CAN Bus Router module.

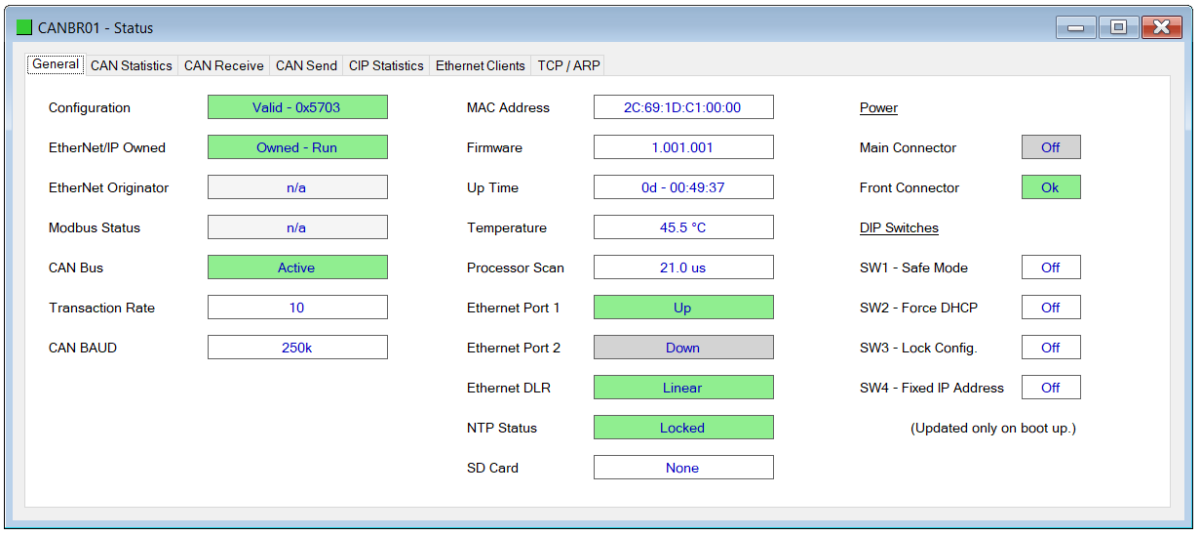


Figure 7.4 - Status monitoring – General

The General tab displays the following general parameters:

Parameter	Description
Configuration	Indicates whether the module's configuration is valid and provides the configuration's CRC checksum.
EtherNet/IP Owned	When the module is configured as an EtherNet/IP Target, this will indicate if the module is owned by an EtherNet/IP connection originator, as well as the Run/Program state of the controller.
EtherNet/IP Originator	When the module is configured as an EtherNet/IP Originator, this will show if all the Class 1 and Explicit Message connections to EtherNet/IP target devices are established and returning valid data.
Modbus Status	When the module is operating as a **Modbus Server**, this parameter will indicate that the module has received a valid Modbus request within the Modbus ***Inactivity Timeout*** time. When the module is operating as a **Modbus Client**, this parameter will indicate that all the mapping items in the Modbus Auxiliary Map are executing correctly.
CAN Bus	The state of the CAN Bus. At least one valid CAN Bus packet must be received within the ***CAN Bus Active Timeout*** time.
Transaction Rate	Number of CAN Bus transactions executed per second.
CAN BAUD	Current Baud Rate of the CAN Bus. Note: This may differ from the configured CAN BAUD rate if Dynamic BAUD Rate Change is used.
MAC Address	Displays the module's unique Ethernet MAC address.
Firmware	The version of the module's firmware.

Up Time	Indicates the elapsed time since the module was powered-up.
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 Port 1/2	This is the status of each Ethernet port. **Down** The Ethernet connector has **not been** successfully connected to an Ethernet network. **Up** The Ethernet connector has successfully connected to an Ethernet network. **Mirror Enabled** The Ethernet port is mirroring the traffic on the other Ethernet port.
Ethernet DLR (Device Level Ring)	The status of the Ethernet DLR. **Disabled** Device Level Ring functionality has been disabled. **Linear** The DLR functionality has been enabled and the Ethernet network architecture is linear. **Ring – Fault** The DLR functionality has been enabled and the Ethernet network architecture is ring, but there is a fault with the network. **Ring – Ok** The DLR functionality has been enabled and the Ethernet network architecture is ring and is operating as expected.
NTP Status	The status of the local NTP Client. **Disabled** The NTP time synchronization has been disabled. **Locked** NTP time synchronization has been enabled and the module has locked onto the target time server. **Not Locked** NTP time synchronization has been enabled and the module has not locked onto the target time server.
SD Card	Indicates if a SD Card is present or not.
Power	Indication from which port the module is receiving power. **Main Connector** The power is present at the bottom connector. **CAN Connector** The power is present at the CAN connector.
DIP Switch Position	The status of the DIP switches when the module booted.

	NOTE: This status will not change if the DIP switches are altered when the module is running.

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

7.2.2. CAN STATISTICS

The CAN Statistics tab displays the statistics associated with the CAN Bus communication network.

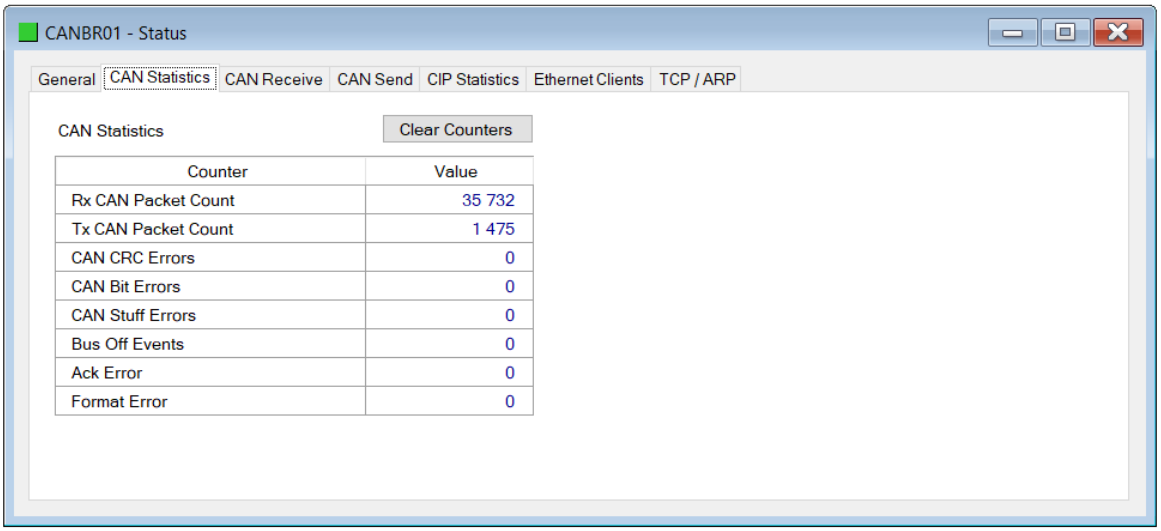


Figure 7.5 - Status monitoring – CAN Bus Statistics

Statistic	Description
Rx CAN Packet Count	The number of CAN packets received.
Tx CAN Packet Count	The number of CAN packets sent.
CAN CRC Errors	The number of received packets where the packet checksum does not match the calculated packet checksum. This implies one or more bits in the frame have been corrupted. May indicate an intermittent CAN cable connection or induced electrical noise.
CAN Bit Errors	The number of transmitted bits where the transmitted bit state does not match the instantaneous read-back state. This may indicate that another device is transmitting at the same time, or one of the CAN lines shorted to power, shorted together, or incorrectly termination.
CAN Stuff Errors	The number of frames received where the required inserted (opposite) bit was not received after 5 identical bits. This may be an indication of bus noise, bad physical cable connection, or a faulty device.
Bus Off Events	The number of Bus-Off Events. A node will enter the Bus-Off state when the transmit Error Count exceeds a certain threshold (typically 125). This may indicate a cable break or loss of power causing the Scanner to enter the Bus-Off state.

Ack Error	The number of transmitted bits that are not read-back and acknowledged by at least one other node. Typically seen when the device is alone on the bus, and there is no other node to acknowledge the frame.
Format Error	The number of received frames where the fixed format part of a received frame is invalid, or the Frame structure is non-standard. (Frame size/type start delimiter etc.) May indicate an intermittent CAN cable connection, induced electrical noise or a node present with an incorrect BAUD rate.

Table 7.3 – CAN statistics

7.2.3. CAN Receive Status

The CAN Receive Status tab displays the associated count for each configured CAN Receive item.

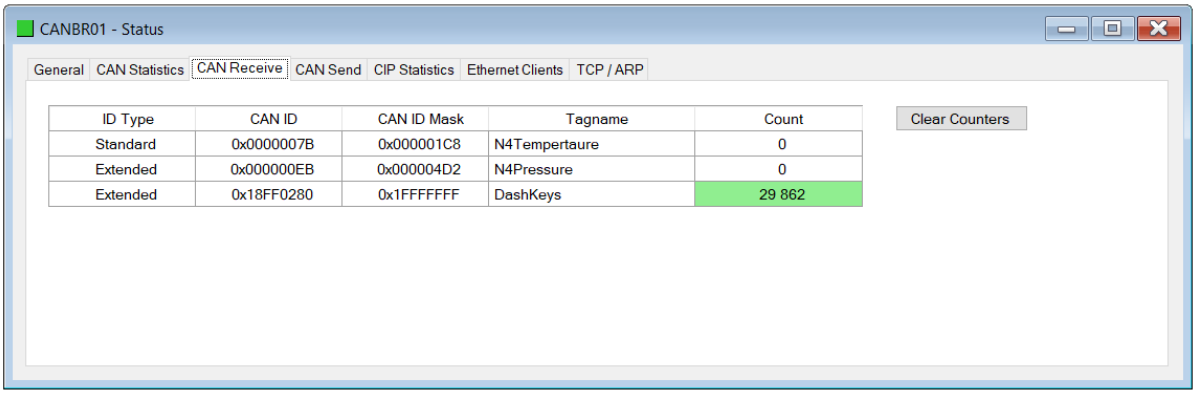


Figure 7.6 - Status monitoring – CAN Receive Status

Statistic	Description
CAN ID Type	The configured CAN ID Type, either: Standard – (11-bit CAN Header) Extended – (29-bit CAN Header)
CAN ID	The configured CAN ID required in a received message to be matched with this transaction.
CAN ID Mask	The configured CAN ID Mask to be used (bitwise-AND) when matching a received message.
Tagname	The configured user defined Tag or Description string for the CAN ID.
Count	The number of received CAN packets that matched this item.

Table 7.4 – CAN Receive status

7.2.4. CAN SEND STATUS

The CAN Send Status tab displays the associated count for each configured CAN Send item.

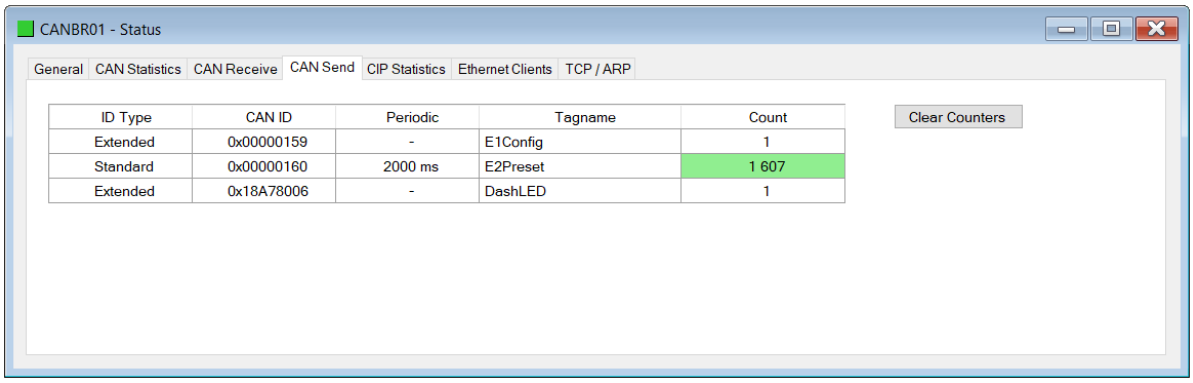


Figure 7.7 - Status monitoring – CAN send Status

Statistic	Description
CAN ID Type	The configured CAN ID Type, either: Standard – (11-bit CAN Header) Extended – (29-bit CAN Header)
CAN ID	The CAN ID of the transmitted CAN packet.
Periodic	The configured periodic interval, if applicable.
Tagname	The configured user defined Tag or Description string.
Count	The number of times this CAN packet has been transmitted.

Table 7.5 – CAN Send status

7.2.5. ETHERNET/IP EXPLICIT

The EtherNet/IP Explicit Statistics tab displays the statistics associated with EtherNet/IP Device explicit mapping.

NOTE: This tab is only applicable when the module has the Primary Interface set to EtherNet/IP Originator.

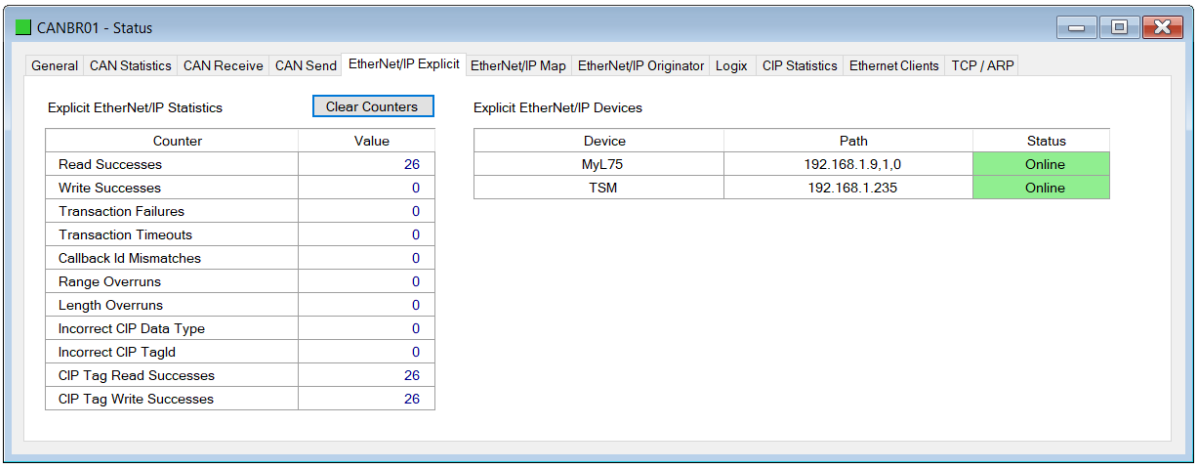


Figure 7.8 - Status monitoring – EtherNet/IP Explicit

Statistic	Description
Read Successes	The number of successful reads from the target EtherNet/IP device.
Write Successes	The number of successful write to the target EtherNet/IP device.
Transaction Failures	The number of failed reads/writes to the target EtherNet/IP device (e.g. error response).
Transaction Timeouts	The number of times the target EtherNet/IP device failed to respond.
Callback Id Mismatches	The EtherNet/IP UCMM or Class 3 response does not match the request.
Range Overruns	The number of times the returned data amount runs over the max Internal Data Space.
Length Overruns	The number of times the returned data is greater than the configured get length.
Incorrect CIP Data Type	When the Explicit Message Function is a Tag Read/Write, this statistic will increase when the incorrect CIP data type was returned in the response.
Incorrect CIP Tag Id	When the Explicit Message Function is a Tag Read/Write, this statistic will increase when the incorrect CIP UDT tag ID was returned in the response.
CIP Tag Read Successes	When the Explicit Message Function is a Tag Read, this statistic will increase when there was a successful Logix Tag Read.
CIP Tag Write Successes	When the Explicit Message Function is a Tag Write, this statistic will increase when there was a successful Logix Tag Write.

Table 7.6 – EtherNet/IP Explicit Statistics

7.2.6. ETHERNET/IP MAP

The EtherNet/IP Map tab displays the success counts for each EtherNet/IP device mapped item.

NOTE: This tab is only applicable when the module has the Primary Interface set to EtherNet/IP Originator.

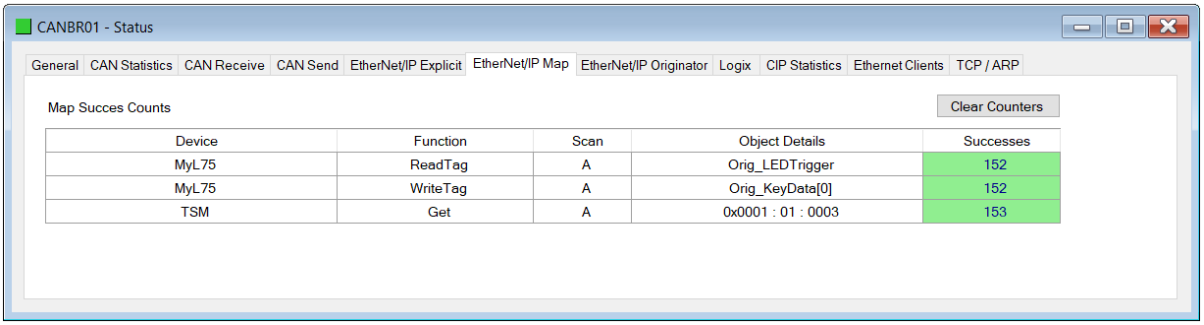


Figure 7.9 - Status monitoring – EtherNet/IP Map

Each time a mapped item is executed successfully its associated count will increase. The count cell will momentarily be highlighted green following a successful transaction.

7.2.7. ETHERNET/IP ORIGINATOR

The EtherNet/IP Originator tab displays the EtherNet/IP Class 1 connection status and statistics for each configured EtherNet/IP device.

NOTE: This tab is only applicable when the module has the Primary Interface set to EtherNet/IP Originator.

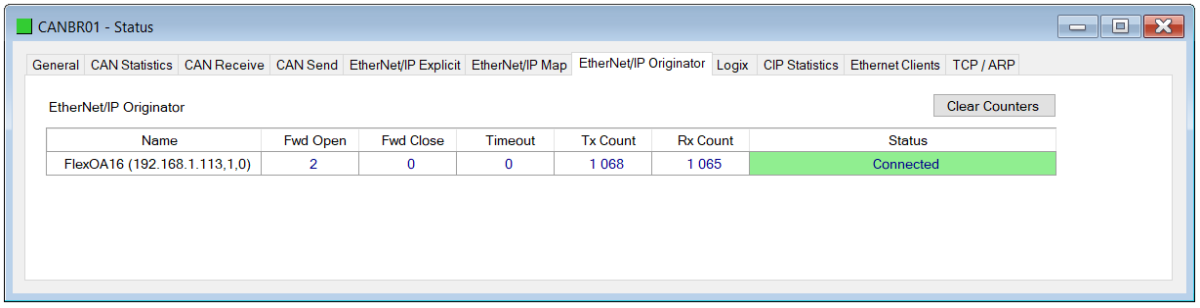


Figure 7.10 - Status monitoring – EtherNet/IP Originator

Statistic	Description
Status	The current connection status of the module.

	Connected The device is connected and exchanging data using Class 1 cyclic communication. **Offline** The device it offline and not connected **Various response faults** If the connection parameters entered are not correct, then generally the target device will reply with the specific reason for the connection reject, for example: Ownership Conflict Connection In Use Or Duplicate Forward Open
Class 1 Originator Statistics	
Forward Open Count	The number of Class 1 Forward Open (connection establishment) messages sent to this device.
Forward Close Count	The number of Class 1 Forward Close (connection termination) messages sent or received from this device.
Connection Timeouts	The number of this connection was closed due to timeouts.
Tx Count	Number of Class 1 messages sent to the specific target device.
Rx Count	Number of Class 1 messages received from the specific target device.

Table 7.7 – EtherNet/IP Class 1 status and statistics

7.2.8. LOGIX

The Logix tab displays the Logix statistics for the explicit EtherNet/IP Tag Read/Write message instructions.

NOTE: This tab is only relevant when the module has the Primary Interface set to EtherNet/IP Originator and Logix Tag Read/Write functions are being used in the EtherNet/IP Explicit Message Map.

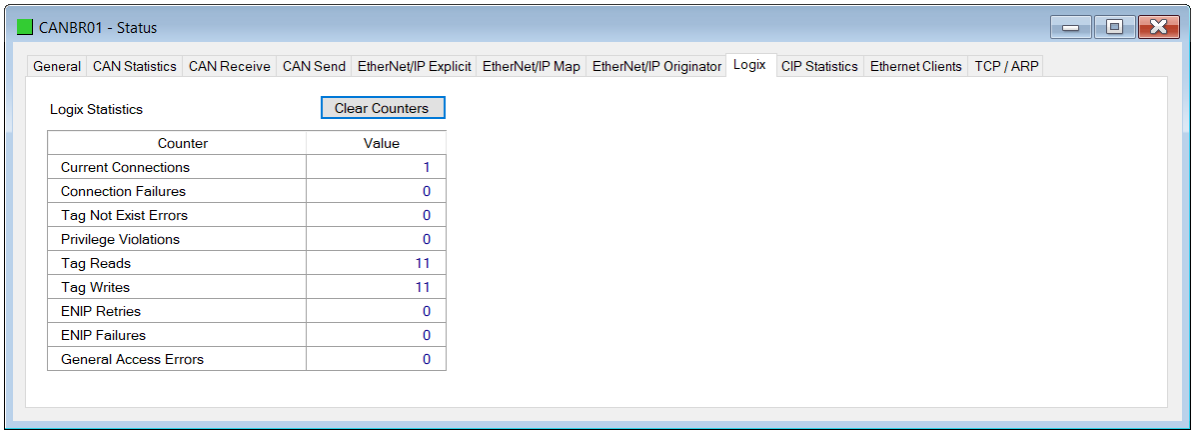


Figure 7.11 - Status monitoring – Logix Statistics

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. Note: 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 CAN Bus Router module.
Tag Writes	The number of tag write transactions executed by the CAN Bus Router module.
ENIP Retries	This count increases when no response is received from the Logix Controller within the ENIP timeout.
ENIP Failures	This count increases when the ENIP Retry Limit is reached and no response has been received from the Logix Controller.
Access General Error	This count increases when a tag cannot be accessed for any other reason not reported above.

Table 7.8 – Logix Statistics Tab

7.2.9. MODBUS

The Modbus tab displays the Modbus statistics for the Modbus Read and Write Message Exchanges when the module is a Modbus TCP Server or Modbus TCP Client.

NOTE: The Modbus statistics tab is only displayed if the module has the primary interface set to Modbus TCP Client or Modbus TCP Server.

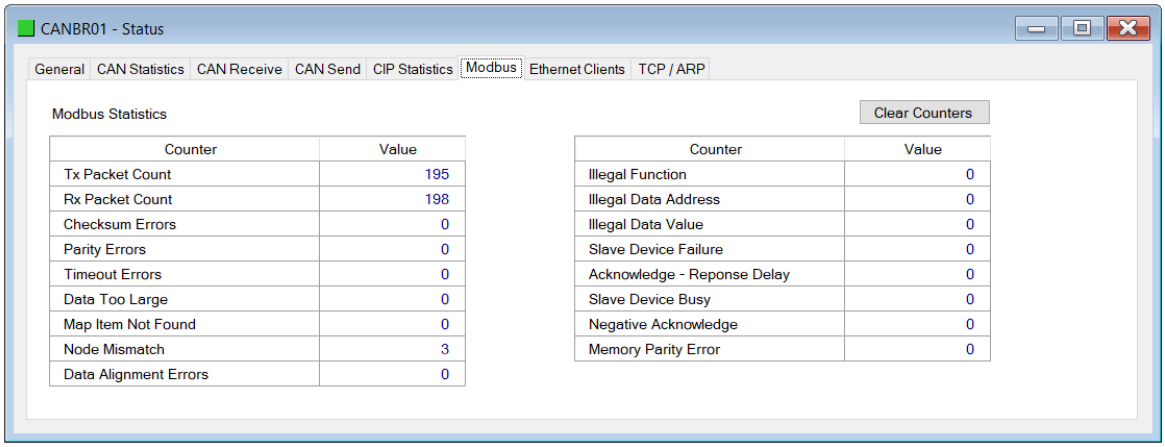


Figure 7.12. - Status monitoring – Modbus Statistics

The Modbus tab displays the following parameters:

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

Negative Acknowledge	The number of times the Modbus device responded with a Negative Acknowledge exception.
Memory Parity Error	The number of times the Modbus device responded with a Memory Parity exception.

Table 7.9 - Modbus Statistics Tab

7.2.10. SERIAL STATISTICS

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

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

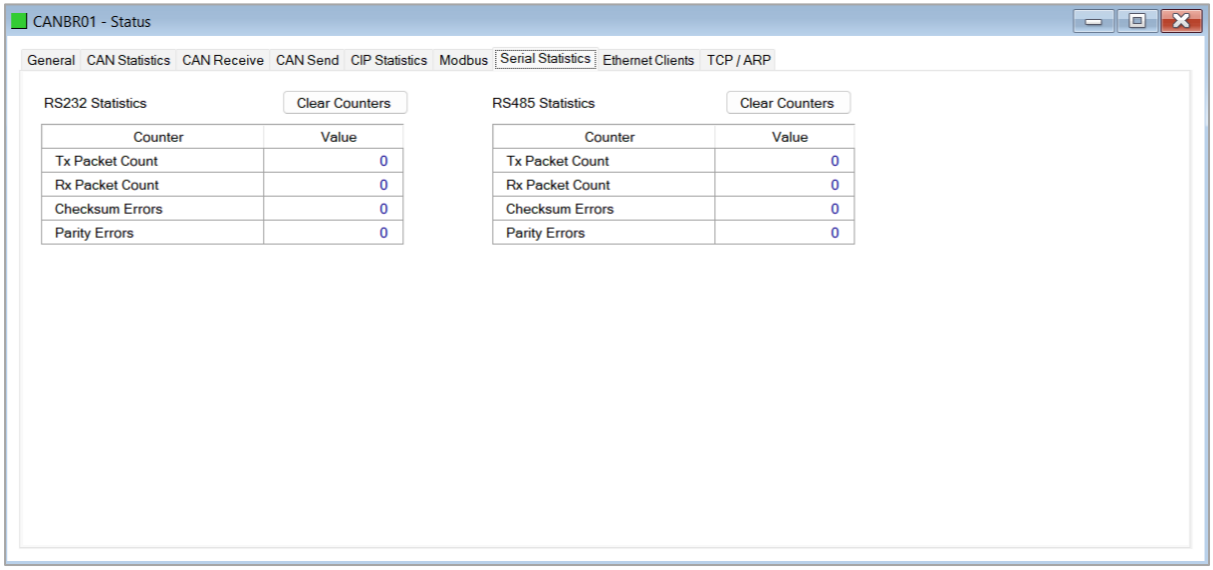


Figure 7.13. - Status monitoring – Serial Statistics

The Serial Statistics tab displays the following parameters:

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

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

Table 7.10 - Serial Statistics Tab

7.2.11. CIP STATISTICS

The CIP tab displays the Ethernet CIP statistics.

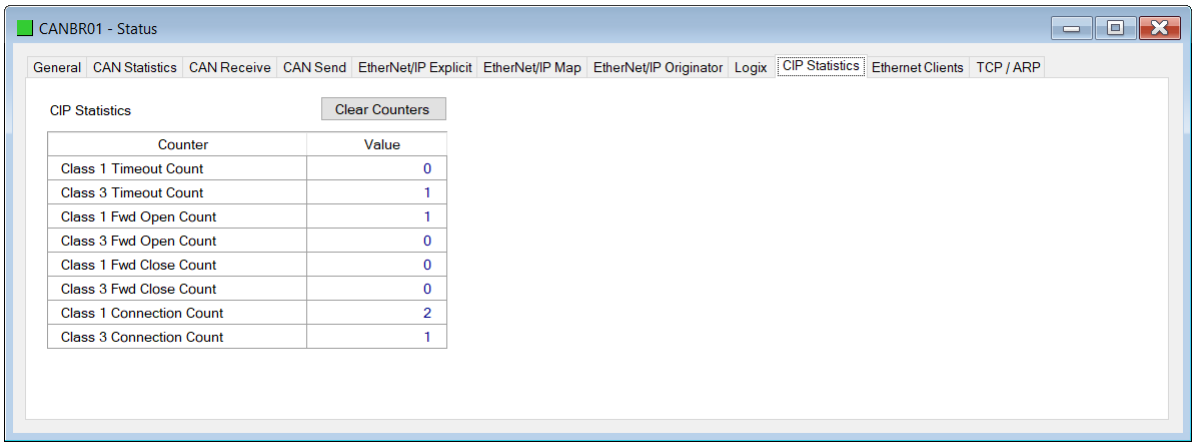


Figure 7.14 - Status monitoring – CIP Statistics

Statistic	**Description**
Class 1 Timeout Count	The number of Class 1 connections closed due to Timeouts.
Class 3 Timeout Count	The number of Class 3 connections closed due to Timeouts.
Class 1 Forward Open Count	The number of Class 1 Forward Open (connection establishment) messages sent.
Class 3 Forward Open Count	The number of Class 3 Forward Open (connection establishment) messages sent.
Class 1 Forward Close Count	The number of Class 1 Forward Close (connection termination) messages sent.
Class 3 Forward Close Count	The number of Class 3 Forward Close (connection termination) messages sent.
Class 1 Connection Count	The current number of active Class 1 connections.
Class 3 Connection Count	The current number of active Class 3 connections.

Table 7.11 – Mapped Item statistics

7.2.12. ETHERNET CLIENTS

The Ethernet Clients tab displays details of the Ethernet and EtherNet/IP clients connected to the CAN Bus Router.

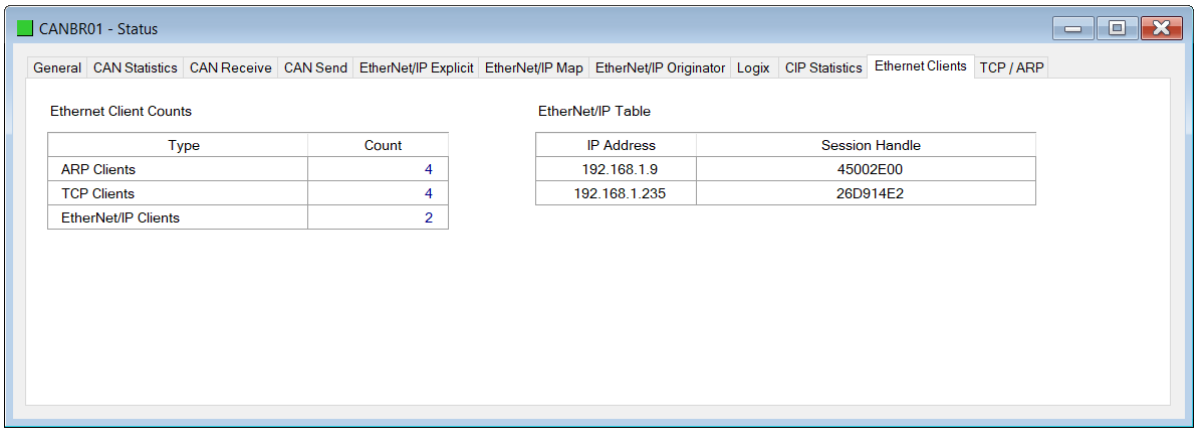


Figure 7.15 –Status monitoring – Ethernet Client Statistics

7.2.13. TCP/ARP

The TCP/ARP tab displays details of the internal Ethernet ARP and TCP lists of the CAN Bus Router.

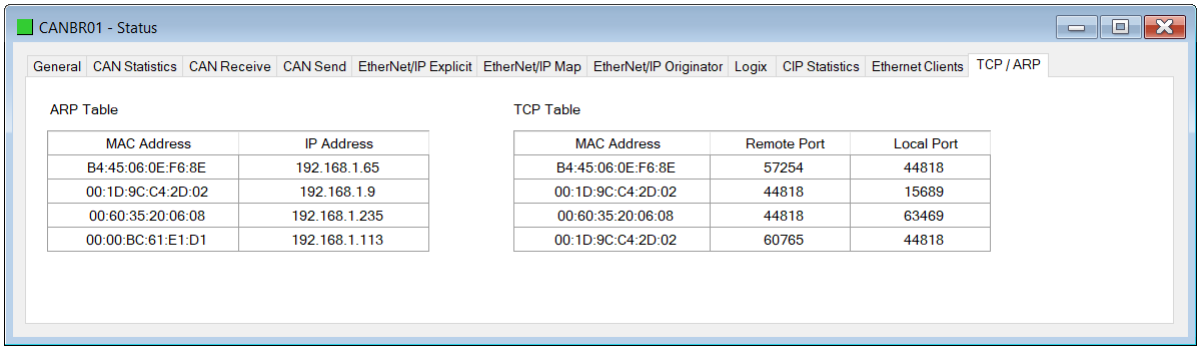


Figure 7.16 – Status monitoring – Ethernet TCP / ARP Statistics

7.3. TARGET DEVICE STATUS MONITORING IN SLATE

The CAN Bus Router can also provide individual statistics and status for each for each of the EtherNet/IP Class 1 or CAN Bus Cyclic IO devices when the Primary Interface is *EtherNet/IP Originator.*

7.3.1. ETHERNET/IP

When online with the module in Slate, right-click on the desired EtherNet/IP device under the *EtherNet/IP Connections* tree in Slate and select *Status*.

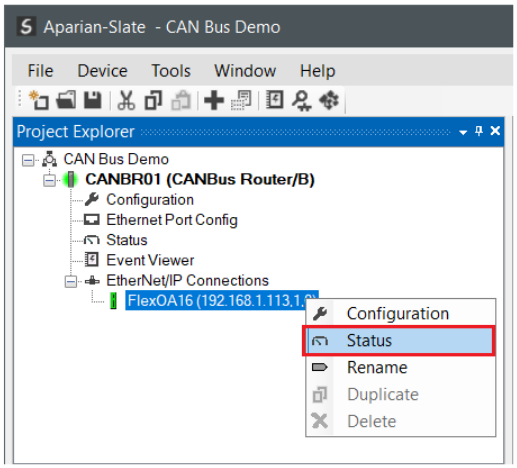


Figure 7.17 – EtherNet/IP Device Status – Status selection

7.3.1.1. GENERAL

The General Status for the EtherNet/IP device shows the connection statistics and parameters associated with the EtherNet/IP Class 1 connection.

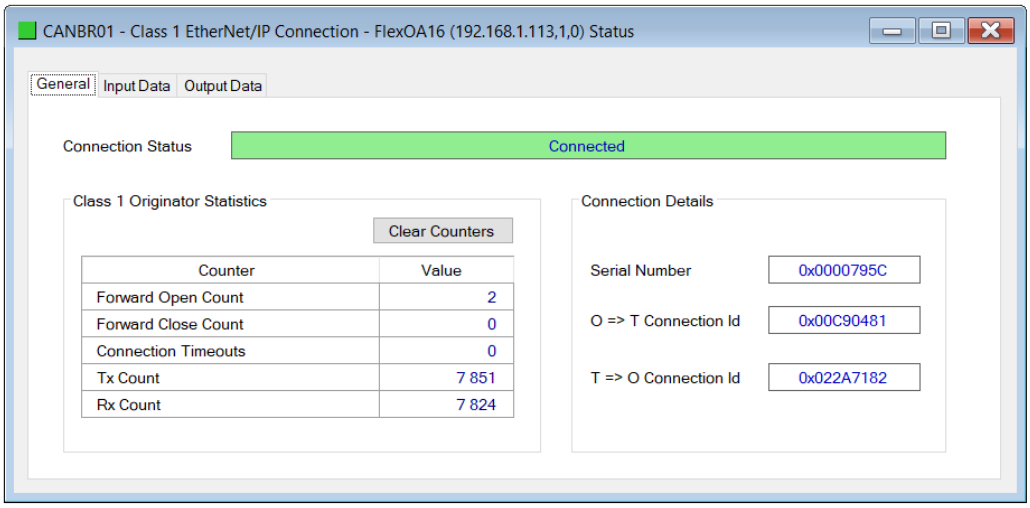


Figure 7.18 – EtherNet/IP Device Status – General Status

Statistic	Description
Connection Status	The current connection status of the module. **Connected** The device is connected and exchanging data using Class 1 cyclic communication. **Offline** The device it offline and not connected **Various response faults** If the connection parameters entered are not correct, then generally the target device will reply with the specific reason for the connection reject, for example: Connection Status: Invalid Originator To Target Size
Class 1 Originator Statistics	
Forward Open Count	The number of Class 1 Forward Open (connection establishment) messages sent to this device.
Forward Close Count	The number of Class 1 Forward Close (connection termination) messages sent or received from this device.
Connection Timeouts	The number of this connection was closed due to timeouts.
Tx Count	Number of Class 1 messages sent to the specific target device.
Rx Count	Number of Class 1 messages received from the specific target device.
Connection Details	
Serial Number	The active connection's serial number.
O -> T Connection Id	The active connection Originator to Target Connection Id.
T -> O Connection Id	The active connection Target to Originator Connection Id.

Table 7.12 – EtherNet/IP Class 1 Device status and statistics

7.3.1.2. INPUT DATA

The Input Data for the EtherNet/IP device shows the Input Assembly associated with the EtherNet/IP Class 1 connection.

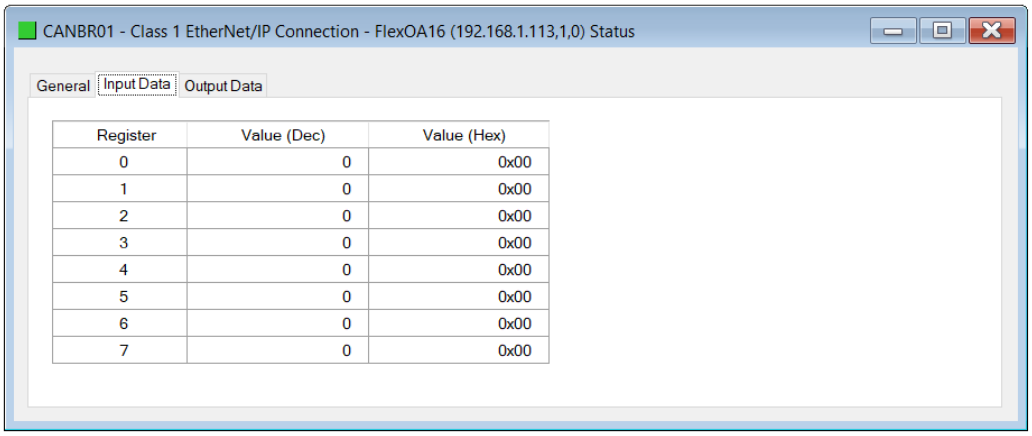


Figure 7.19 – EtherNet/IP Device Status – Input Data

7.3.1.3. OUTPUT DATA

The Output Data for the EtherNet/IP device shows the Output Assembly associated with the EtherNet/IP Class 1 connection.

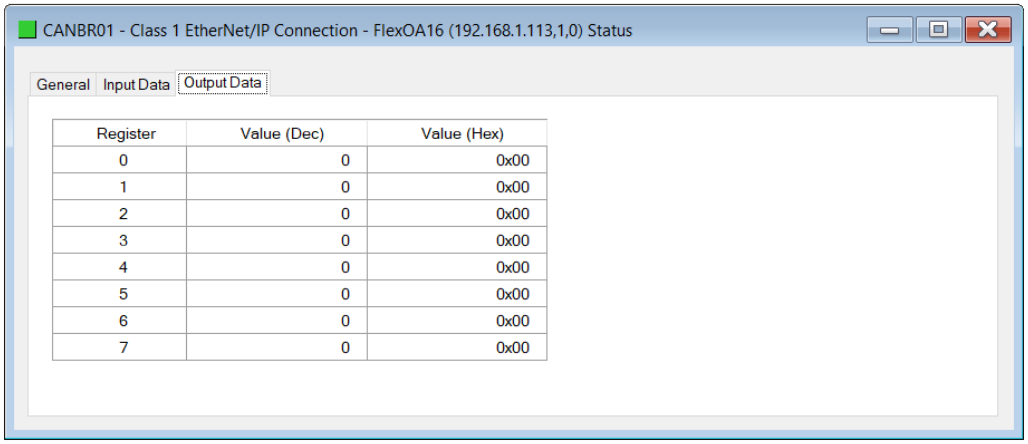


Figure 7.20 – EtherNet/IP Device Status – Output Data

7.4. MODULE EVENT LOG

The CAN Bus 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.

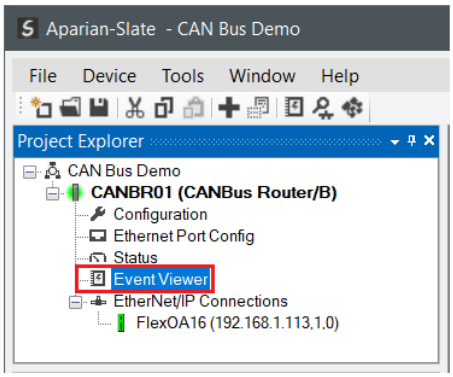
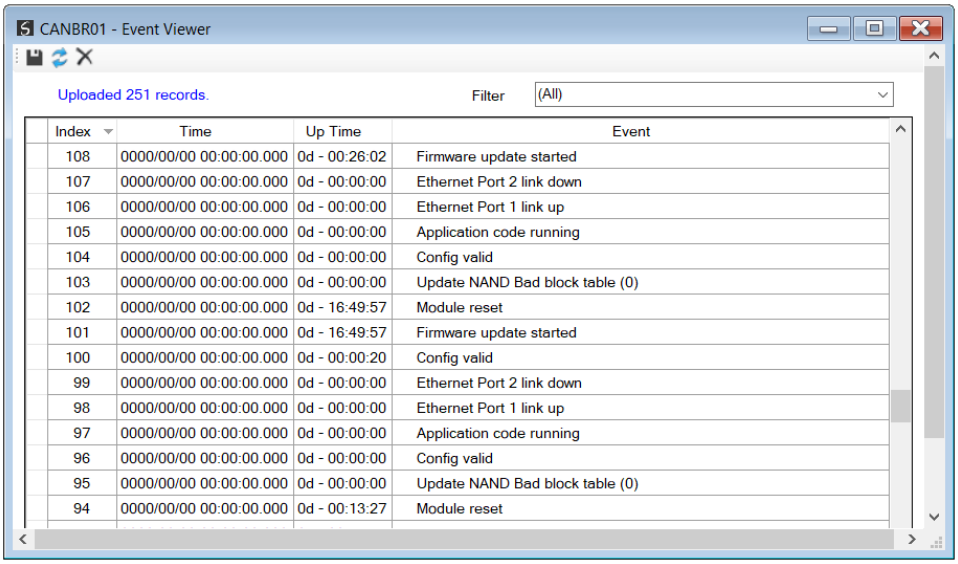


Figure 7.21 - Selecting the module Event Log

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	Time	Up Time	Event
108	0000/00/00 00:00:00.000	0d - 00:26:02	Firmware update started
107	0000/00/00 00:00:00.000	0d - 00:00:00	Ethernet Port 2 link down
106	0000/00/00 00:00:00.000	0d - 00:00:00	Ethernet Port 1 link up
105	0000/00/00 00:00:00.000	0d - 00:00:00	Application code running
104	0000/00/00 00:00:00.000	0d - 00:00:00	Config valid
103	0000/00/00 00:00:00.000	0d - 00:00:00	Update NAND Bad block table (0)
102	0000/00/00 00:00:00.000	0d - 16:49:57	Module reset
101	0000/00/00 00:00:00.000	0d - 16:49:57	Firmware update started
100	0000/00/00 00:00:00.000	0d - 00:00:20	Config valid
99	0000/00/00 00:00:00.000	0d - 00:00:00	Ethernet Port 2 link down
98	0000/00/00 00:00:00.000	0d - 00:00:00	Ethernet Port 1 link up
97	0000/00/00 00:00:00.000	0d - 00:00:00	Application code running
96	0000/00/00 00:00:00.000	0d - 00:00:00	Config valid
95	0000/00/00 00:00:00.000	0d - 00:00:00	Update NAND Bad block table (0)
94	0000/00/00 00:00:00.000	0d - 00:13:27	Module reset

Figure 7.22 - Module Event Log

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

7.5. WEB SERVER

The CAN Bus Router provides a web server allowing a user without Slate to view various diagnostics of the module. This includes Ethernet parameters, system event log, advanced diagnostics, and application diagnostics (CAN Bus diagnostics).

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

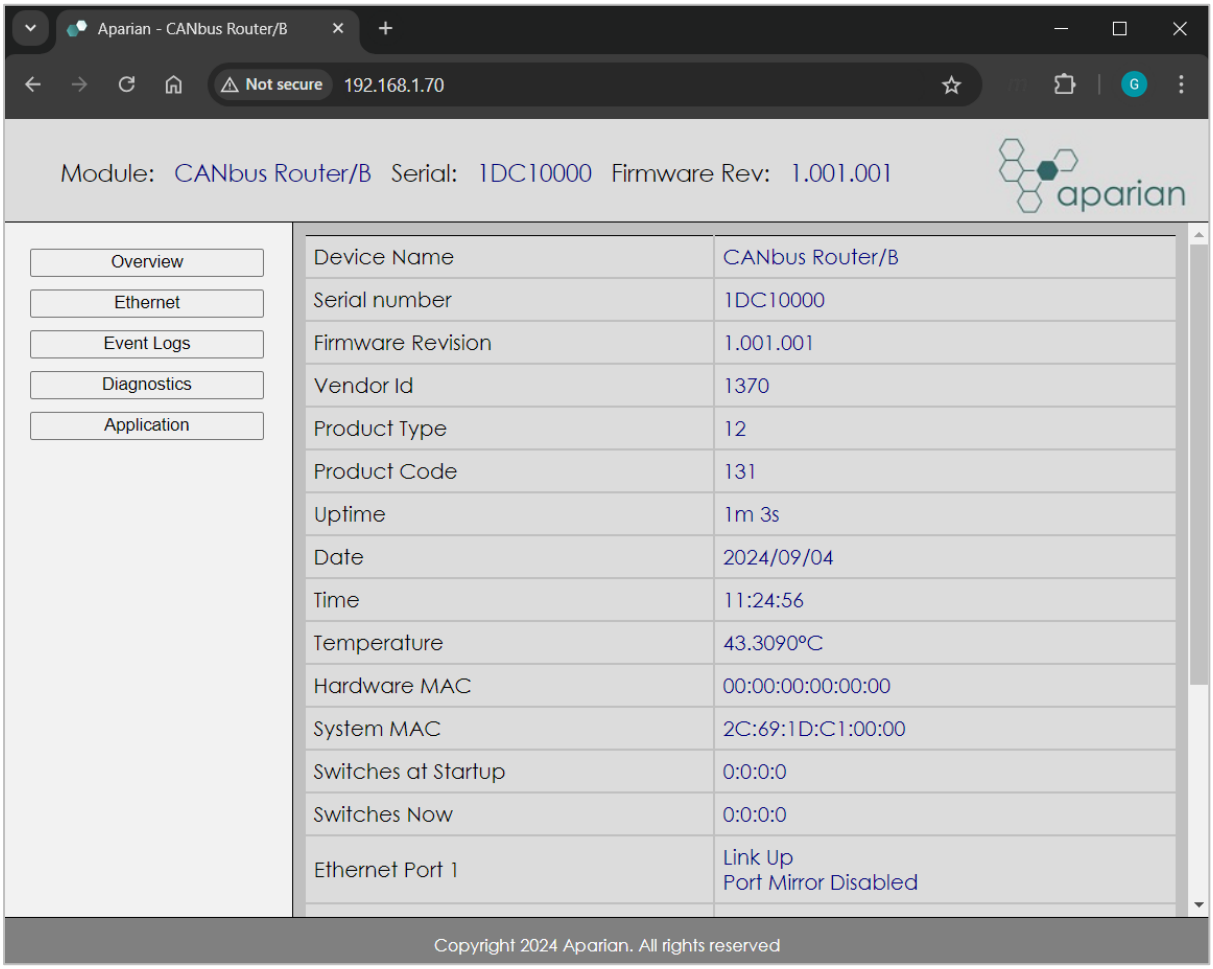


Figure 7.23 – Web interface

7.6. CAN BUS PACKET CAPTURE

The module provides the capability to capture the CAN Bus traffic for analysis. This 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 CAN Bus Packet Capture item in the Project Explorer tree.

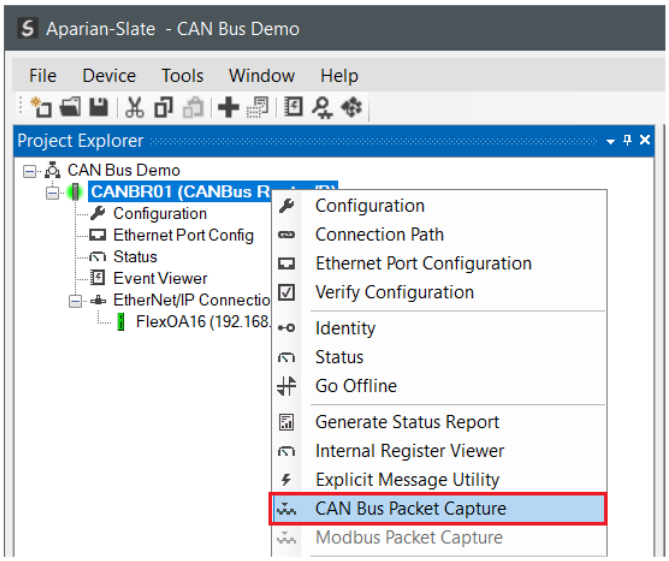


Figure 7.24 - Selecting CAN Bus Packet Capture

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

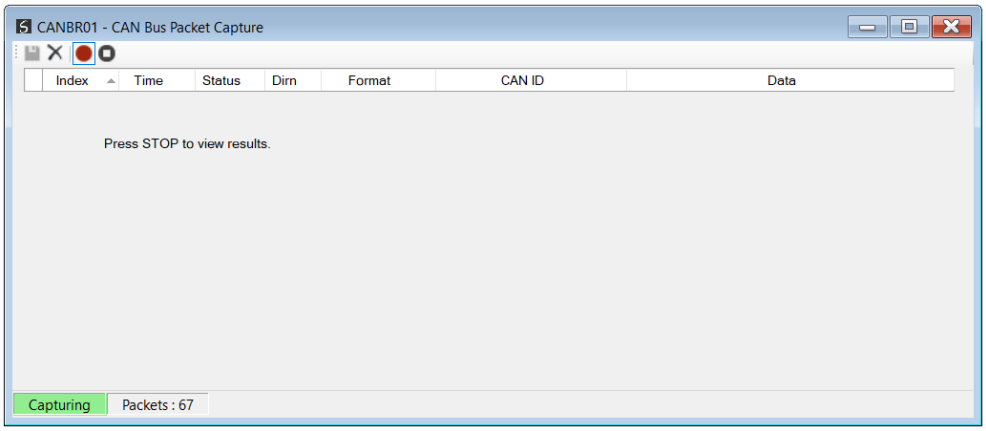


Figure 7.25 – CAN Bus packet capture

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

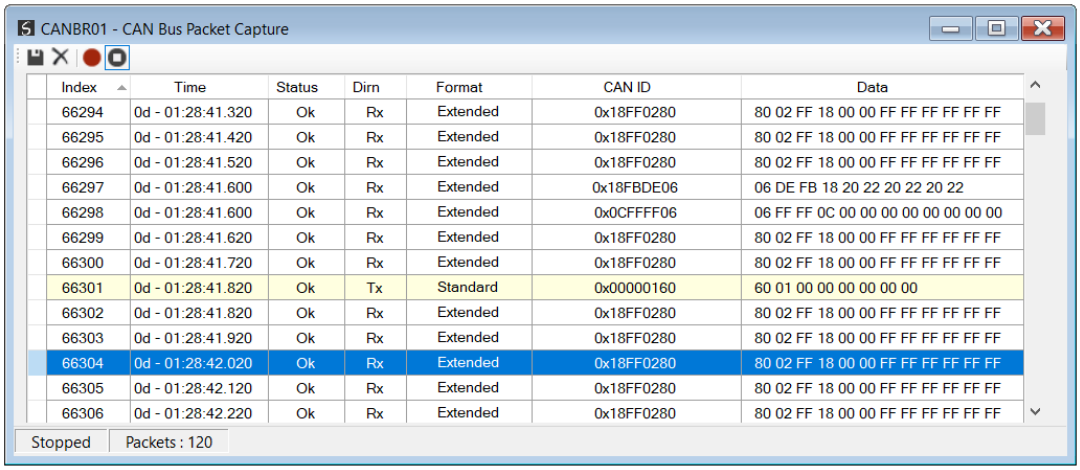


Figure 7.26 – CAN Bus Packet Capture complete

The captured CAN Bus 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 CAN Bus constructs and valid checksums.
Dirn	The direction of the packet, either transmitted (Tx) or received (Rx).
Format	The format of the CAN header either: Standard: (11-bit) Extended: (29-bit)
CAN ID	The CAN ID of the packet.
Data	The raw packet data.

Table 7.13 – CAN Bus 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 CAN Bus Packet Capture files can be viewed by selecting the ***CAN Bus Packet Capture Viewer*** option in the ***Tools*** menu.

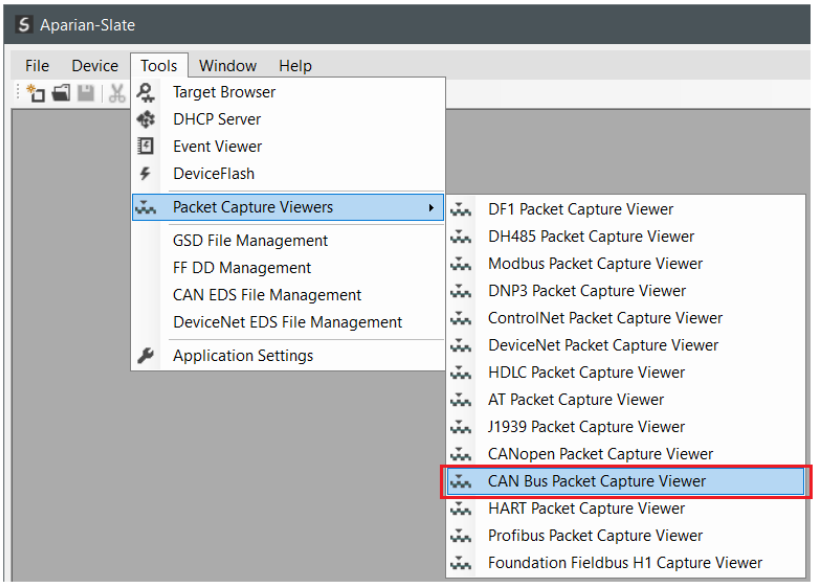


Figure 7.27 - Selecting the CAN Bus Packet Capture Viewer

7.7. MODBUS PACKET CAPTURE

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

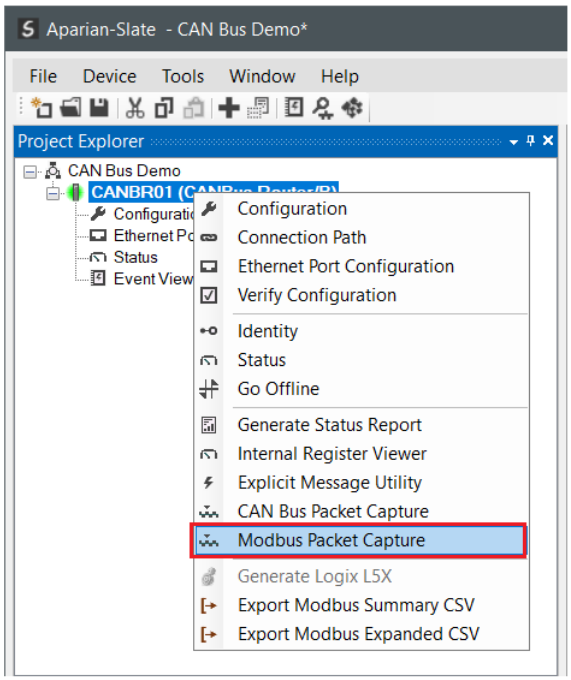


Figure 7.28 - Selecting Modbus Packet Capture

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

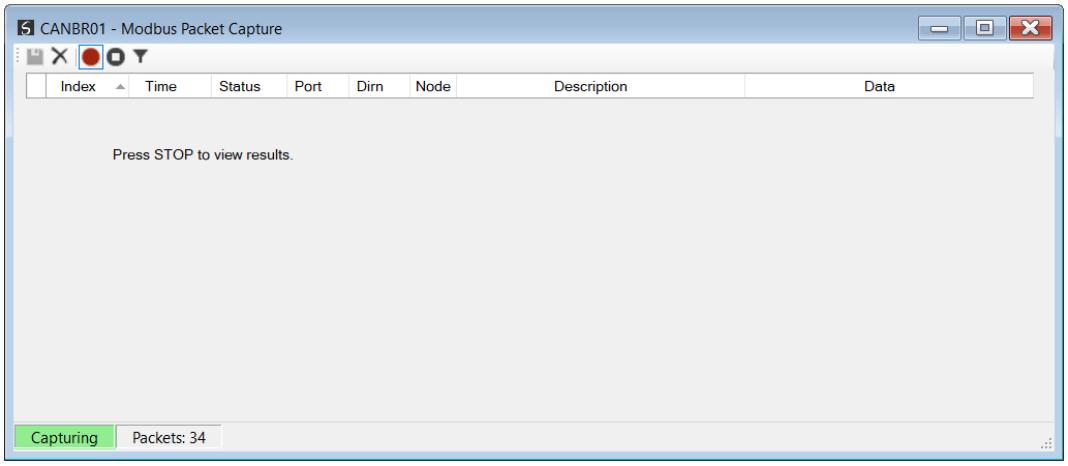


Figure 7.29 – Modbus packet capture

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

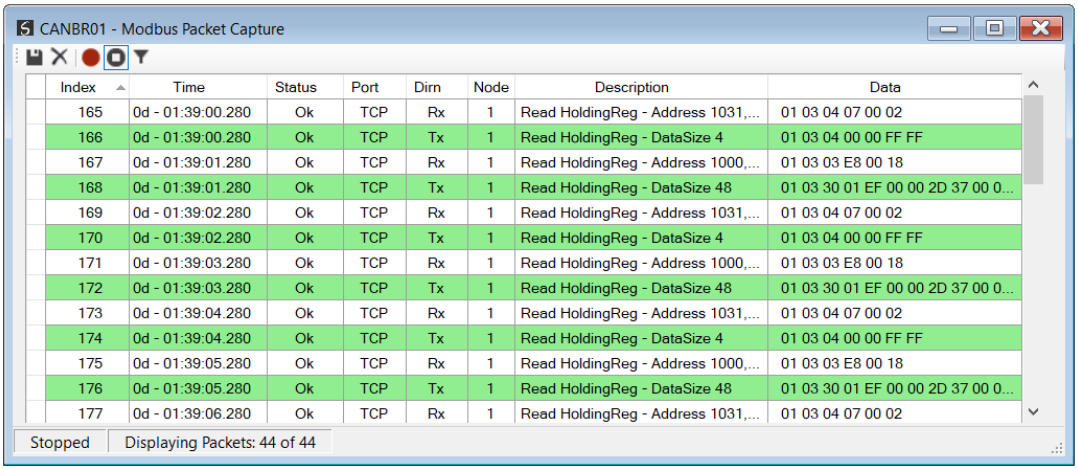


Figure 7.30 – Modbus Packet Capture complete

The captured Modbus packets are tabulated as follows:

Statistic	Description
Index	The packet index, incremented for each packet sent or received.
Time	The elapsed time since the module powered up.
Status	The status of the packet. Received packets are checked for valid Modbus constructs and valid checksums.

Port	Port on where the data was sent or received (TCP, RTU232, RTU485)
Dirn	The direction of the packet, either transmitted (Tx) or received (Rx).
Node	The Source Node address for the packet
Description	Description of the packet that was received.
Data	The raw packet data.

Table 7.14 – Modbus Packet Capture fields

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

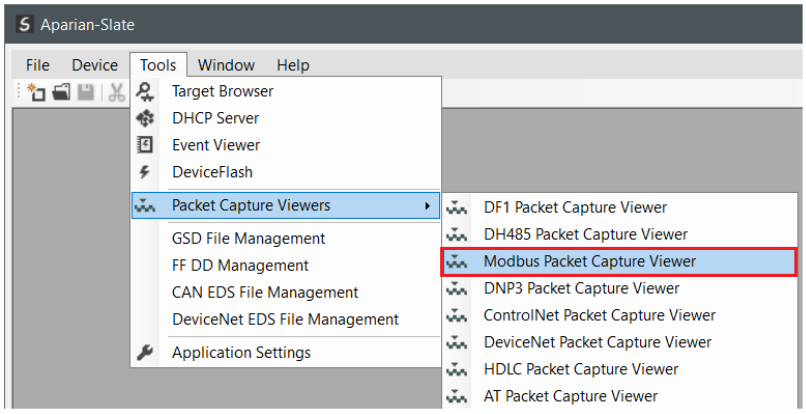


Figure 7.31 - Selecting the Modbus Packet Capture Viewer

7.8. MODULE STATUS REPORT

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

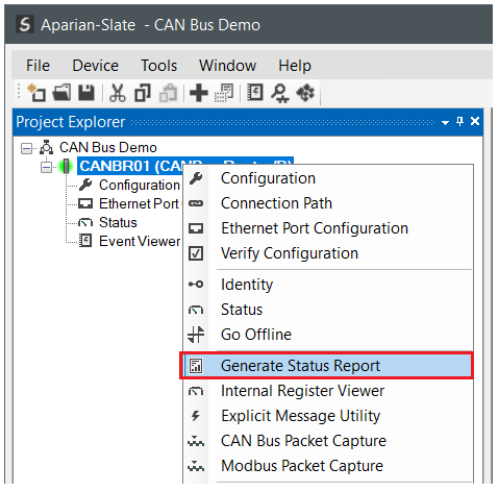


Figure 7.32 – Module Status Report

7.9. MODBUS SUMMARY CSV

Slate can provide a summary of the Modbus registers being used in the form of a CSV file. This will assist the user to better understand where what data is being mapped (based on the CAN Bus Router module configuration). To generate the Modbus Summary CSV, right-click on the module (when online in Slate), and select *Export Modbus Summary CSV*.

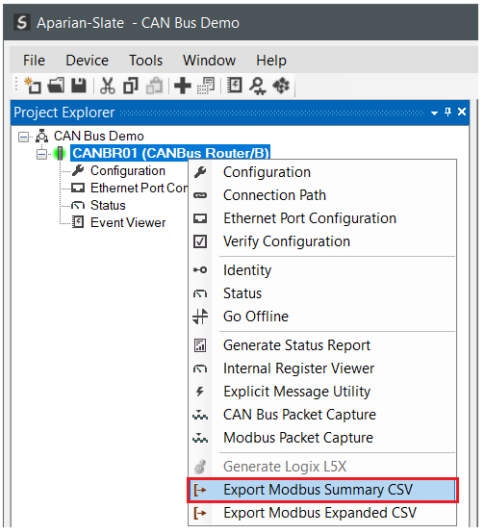


Figure 7.33 – Modbus Summary Report Generation

	A	B	C	D	E	F
1	Reg Type	Modbus Function	Start Address	End Address	Element Count	Description
2	HR	Read	1000	1023	24	System.Status
3	HR	Read	1024	1024	1	Internal.Data - N4Tempertaure.Data
4	HR	Read	1025	1026	2	Internal.Data - N4Tempertaure.Count
5	HR	Read	1027	1028	2	Internal.Data - N4Pressure.Data
6	HR	Read	1029	1030	2	Internal.Data - N4Pressure.Count
7	HR	Read	1031	1034	4	Internal.Data - DashKeys.Data
8	HR	Read	1035	1036	2	Internal.Data - DashKeys.Count
9	HR	Write	1037	1038	2	Internal.Data - E1Config.Data
10	HR	Write	1039	1040	2	Internal.Data - E1Config.Trigger
11	HR	Read	1041	1042	2	Internal.Data - E1Config.Count
12	HR	Write	1043	1045	3	Internal.Data - E2Preset.Data
13	HR	Read	1046	1047	2	Internal.Data - E2Preset.Count
14	HR	Write	1048	1051	4	Internal.Data - DashLED.Data
15	HR	Write	1052	1053	2	Internal.Data - DashLED.Trigger
16	HR	Read	1054	1055	2	Internal.Data - DashLED.Count
17	(End)					

Figure 7.34 – Modbus Summary CSV

7.10. MODBUS EXPANDED CSV

Slate can provide a detailed version of the Modbus registers being used in the form of a CSV file. This will assist the user to better understand where what data is being mapped (based on the CAN Bus Router module configuration). To generate the Modbus Expanded CSV, right-click on the module (when online in Slate), and select *Export Modbus Expanded CSV*.

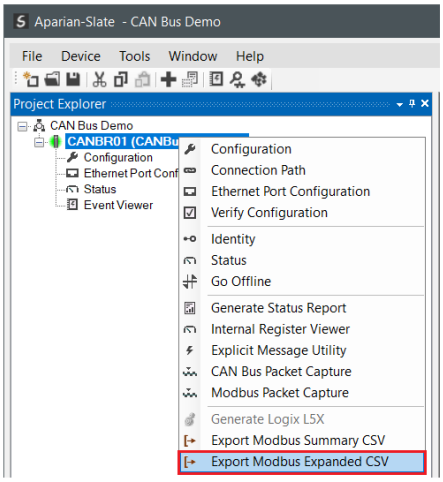


Figure 7.35 – Modbus Expanded Report Generation

	A	B	C	D	E	F
1	Reg Type	Modbus Function	Start Address	End Address	Element Count	Description
2	HR	Read	1000	1001	2	RouterStatus.GeneralStatus
3	HR	Read	1002	1002	1	RouterStatus.ConfigCRC
4	HR	Read	1003	1003	1	RouterStatus.TransactionRate
5	HR	Read	1004	1004	1	RouterStatus.CurrentBaudRate
6	HR	Read	1006	1007	2	RouterStatus.Temperature
7	HR	Read	1008	1009	2	RouterStatus.RxCanPacketCount
8	HR	Read	1010	1011	2	RouterStatus.TxCanPacketCount
9	HR	Read	1012	1013	2	RouterStatus.CANCRCErrors
10	HR	Read	1014	1015	2	RouterStatus.CANBitErrors
11	HR	Read	1016	1017	2	RouterStatus.CanStuffErrors
12	HR	Read	1018	1019	2	RouterStatus.BusOffCount
13	HR	Read	1020	1021	2	RouterStatus.CANAckErrors
14	HR	Read	1022	1023	2	RouterStatus.CANFormatErrors
15	HR	Read	1024	1024	1	Internal.Data - N4Tempertaure.Data
16	HR	Read	1025	1026	2	Internal.Data - N4Tempertaure.Count
17	HR	Read	1027	1028	2	Internal.Data - N4Pressure.Data
18	HR	Read	1029	1030	2	Internal.Data - N4Pressure.Count
19	HR	Read	1031	1034	4	Internal.Data - DashKeys.Data
20	HR	Read	1035	1036	2	Internal.Data - DashKeys.Count
21	HR	Write	1037	1038	2	Internal.Data - E1Config.Data
22	HR	Write	1039	1040	2	Internal.Data - E1Config.Trigger
23	HR	Read	1041	1042	2	Internal.Data - E1Config.Count
24	HR	Write	1043	1045	3	Internal.Data - E2Preset.Data
25	HR	Read	1046	1047	2	Internal.Data - E2Preset.Count
26	HR	Write	1048	1051	4	Internal.Data - DashLED.Data
27	HR	Write	1052	1053	2	Internal.Data - DashLED.Trigger
28	HR	Read	1054	1055	2	Internal.Data - DashLED.Count
29	(End)					

Figure 7.36 – Modbus Expanded CSV

7.11. INTERNAL REGISTER VIEWER

The module provides an Internal Register Viewer that allows the user to view the data registers in the Internal Data Space (IDS) and Modbus Registers.

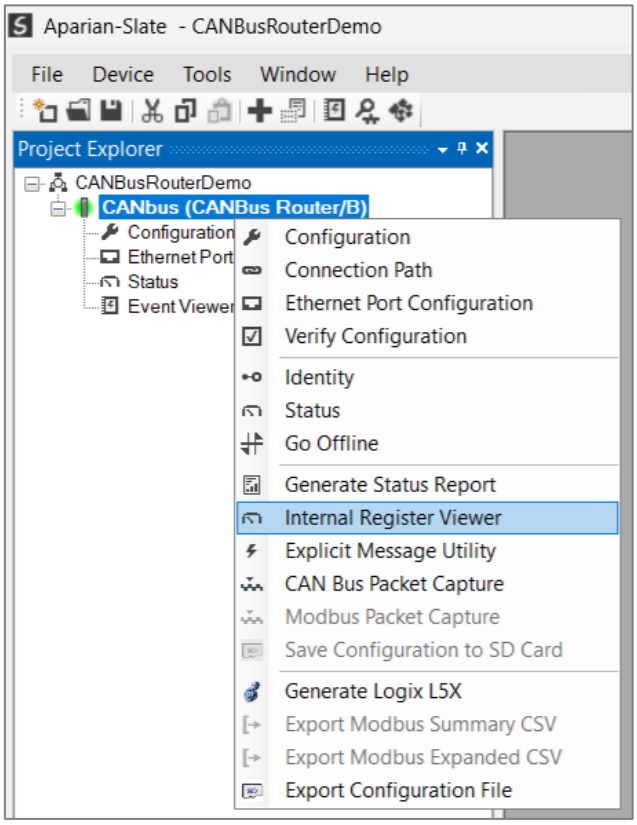


Figure 7.37 – Internal Register Viewer - Select

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

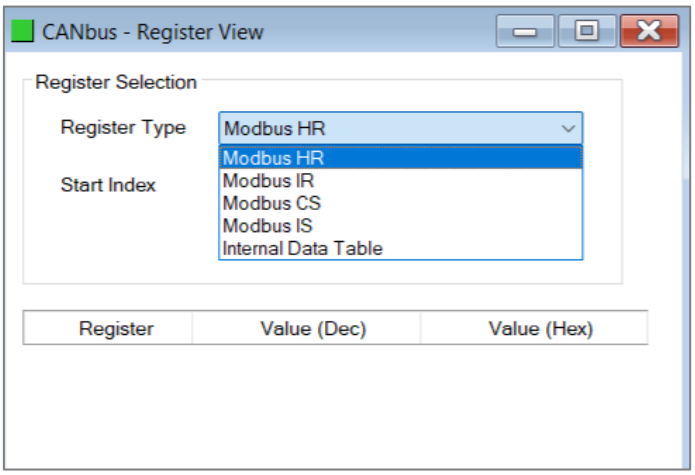


Figure 7.38 – Internal Register Viewer – Register Type

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

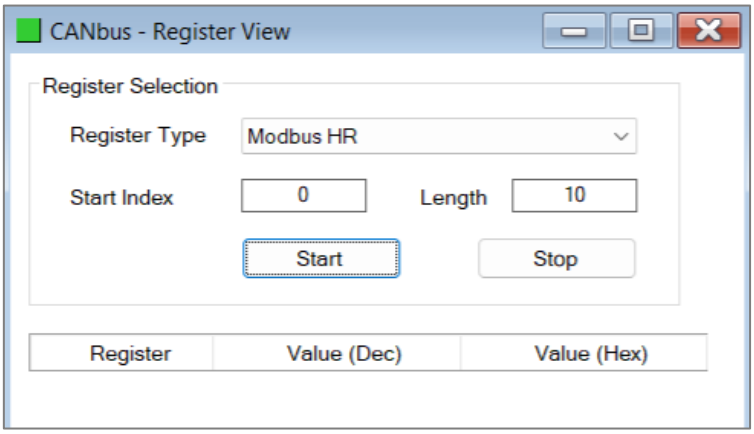


Figure 7.39 – Internal Register Viewer – Indexes

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

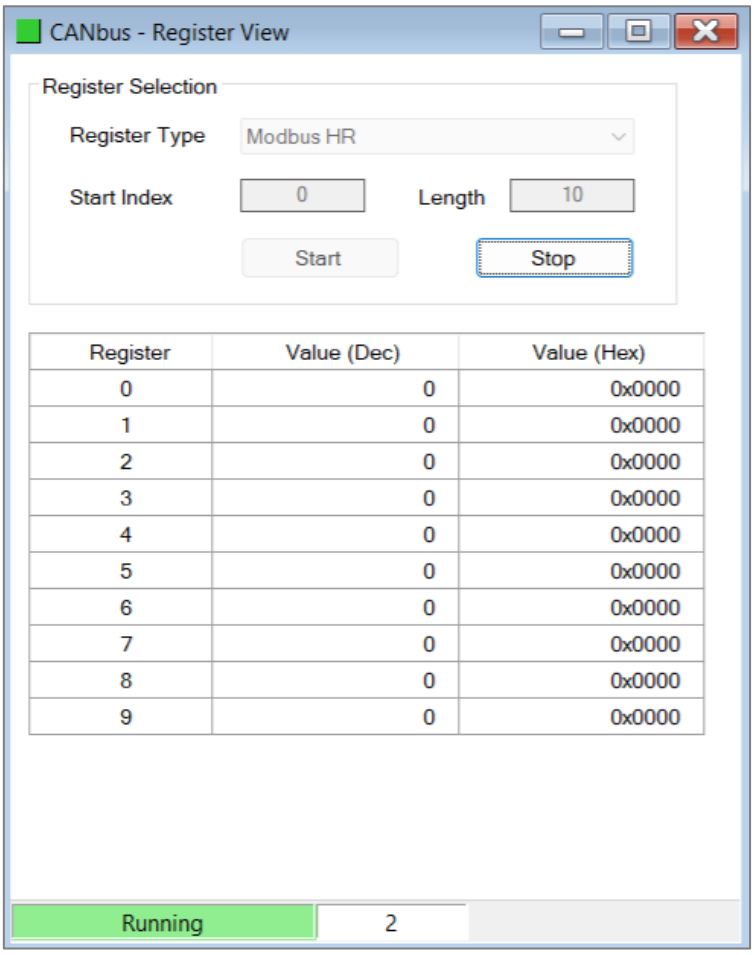


Figure 7.40 – Internal Register Viewer – Running

8. TECHNICAL SPECIFICATIONS

8.1. DIMENSIONS

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

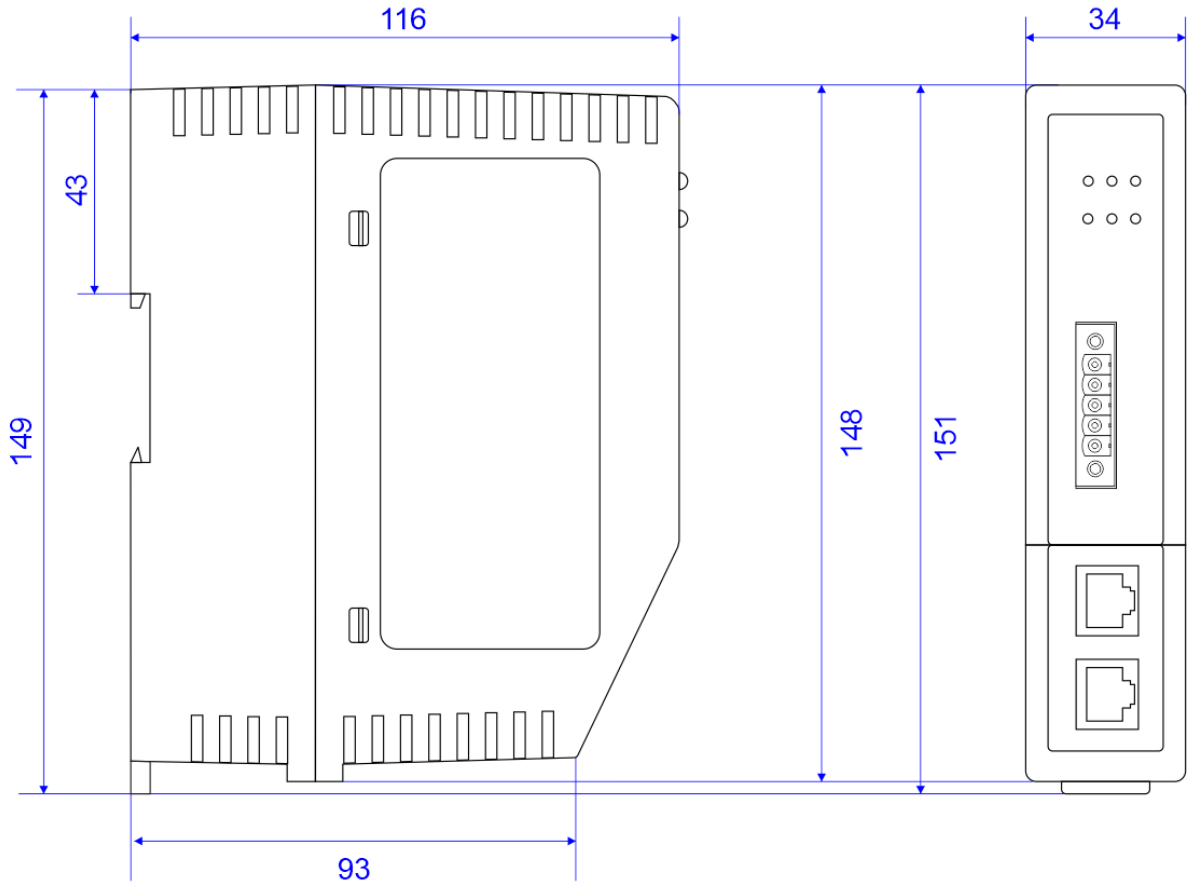


Figure 8.1 – CAN Bus Router enclosure dimensions

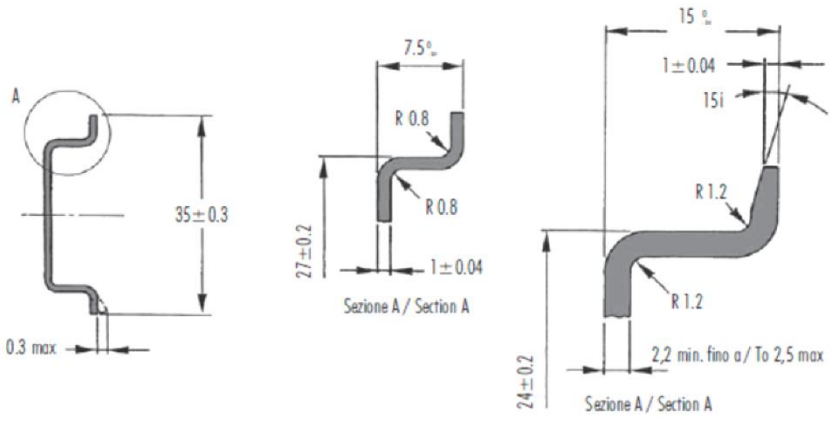


Figure 8.2 - Required DIN dimensions

8.2. ELECTRICAL

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

Table 8.1 - Electrical specification

8.3. ETHERNET

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

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

Table 8.2 - Ethernet specification

8.4. SERIAL PORT (RS232)

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

Table 8.3 – RS232 Serial Port specification

8.5. SERIAL PORT (RS485)

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

Table 8.4 – RS485 Serial Port specification

8.6. CAN BUS

Specification	Rating
Connector	5-way terminal, 5.08mm pitch.
Max CAN Receive Items	100
Max CAN Send Items	100
Passthrough Messaging	Supported
Supported BAUD Rates	10k, 20k, 50k, 125k, 250k, 500k, 800k, 1M
CAN Terminator	120 Ω - Software Enabled

Table 8.5 – CAN Bus specification

8.7. ETHERNET/IP TARGET

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

Table 8.6 – EtherNet/IP Target specification

8.8. ETHERNET/IP ORIGINATOR

Specification	Rating
Class 1 Cyclic Connections Supported	Yes
Class 3 / UCMM Connections Supported	Yes
Class 1 Connection Count	10
Class 3 / UCMM Target Device Count	10
Class 3 / UCMM Mapping Count	50

Table 8.7 – EtherNet/IP Originator specification

8.9. MODBUS CLIENT

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

Table 8.8 – Modbus Client specification

8.10. MODBUS SERVER

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

Table 8.9 – Modbus Server specification

8.11. 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
UKCA	UK CA
ATEX	Ex II 3 G Ex ec IIC T5 -25°C ≤ Ta ≤ 70 °C
ODVA Conformance	EtherNet/IP™

Table 8.10 – Certifications

9. INDEX

A

Adanced, 59, 60

C

CAN BUS, 124
CAN BUS general configuration, 26, 28, 30, 32
CAN BUS parameters, 26
CAN BUS Router, 7, 13, 14, 17, 25, 90, 113
CAN BUS Router, 7
CAN BUS Router, 74
CAN BUS Router, 102
CAN BUS Router, 113
CAN BUS Router, 121
CAN BUS Router, 122
CAN BUS Router general configuration, 43, 45, 46, 48, 63, 65
Connection path, 74
Contact Us, 12

D

DC power, 13
DHCP, 14, 19, 20, 21, 22
dimensions, 133
DIN rail, 15, 16, 133
DIP, 14

E

Ethernet connector, 18
Ethernet/IP, 60
EtherNet/IP, 7, 60
EtherNet/IP Devices configuration, 59
EtherNet/IP Map configuration, 60, 61, 72, 73, 74

F

firmware upgrade, 26

I

Input assembly, 107, 108, 109
input voltage, 16, 18
IP Address, 20, 21

L

LED, 101, 102

M

MODBUS, 114

O

output assembly, 90

R

Rockwell Automation, 24
RS232, 13, 14, 17
RS232/RS485, 17
RSLinx, 24

S

Safe Mode, 14
Serial, 135
Slate, 26, 76, 102, 118, 121, 122
statistics, 111, 112, 113, 117, 119, 124, 127
Statistics, 102
Support email, 12

T

Target Browser, 22, 23, 60, 75

W

web server, 102, 122