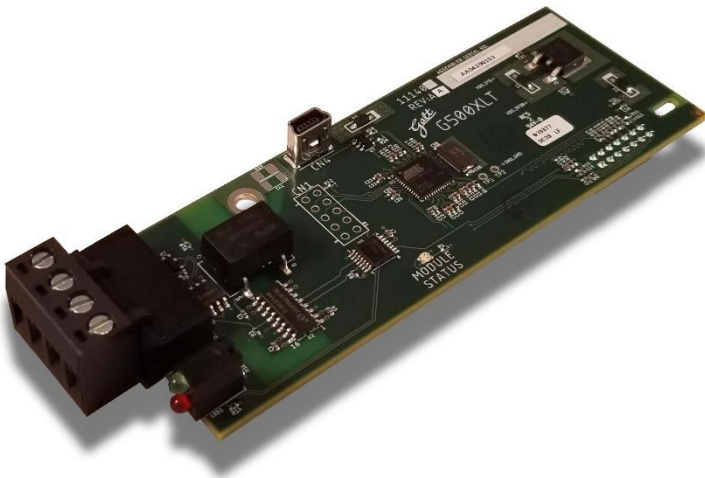




Galt Electric

G500 SERIES

G500XLT Multiprotocol RS-485 Interface



CAUTION

Thank you for purchasing the G500XLT Multiprotocol RS-485 Interface.

- This product is designed to connect the G500 series of inverters to RS-485 communication networks. Please read this instruction manual thoroughly in order to become familiar with the proper interface handling, installation and usage procedures.
- Improper handling may inhibit correct operation or cause premature interface failure.
- Please deliver this instruction manual to the end user of the interface, and retain it in an accessible location.
- For inverter usage instructions, please refer to the applicable inverter instruction manual.



G500XLT Multiprotocol RS-485 Interface Instruction Manual

Part Number 11143

Printed in U.S.A.

©2021 Galt Electric.

All rights reserved

Galt Electric reserves the right to make changes and improvements to its products without providing notice.

Notice to Users

PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE-SUPPORT DEVICES OR SYSTEMS. Life-support devices or systems are devices or systems intended to sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling and user's manual, can be reasonably expected to result in significant injury.

No complex software or hardware system is perfect. Bugs may always be present in a system of any size. In order to prevent danger to life or property, it is the responsibility of the system designer to incorporate redundant protective mechanisms appropriate to the risk involved.

Preface

This instruction manual has been prepared to help you connect your G500 series inverter to RS-485 networks using the G500XLT Multiprotocol RS-485 interface card. This instruction manual does not contain inverter usage instructions. Please refer to this instruction manual in conjunction with the applicable inverter instruction manual in order to become familiar with the proper handling, installation and operation of this product. Improper handling or installation procedures may result in incorrect operation or premature product failure.

Related Publications

Listed below are publications that are necessary for reference in conjunction with this instruction manual.



- **G500 Series Inverter Operation Manual**

These documents are subject to change without notice. Please be sure to refer to the most recent available versions.

Safety precautions

Please read this instruction manual thoroughly prior to proceeding with installation, connections, operation, or maintenance and inspection. Additionally, ensure that all aspects of the system are fully understood, and familiarize yourself with all safety information and precautions before operating the inverter.

Safety precautions in this instruction manual are classified into the following two categories:

 WARNING	Failure to heed the information indicated by this symbol may lead to dangerous conditions, possibly resulting in death or serious bodily injuries.
 CAUTION	Failure to heed the information indicated by this symbol may lead to dangerous conditions, possibly resulting in minor or light bodily injuries and/or substantial property damage.

Failure to heed the information contained under the CAUTION title can also result in serious consequences. These safety precautions are of utmost importance and must be observed at all times.

Installation and Wiring

WARNING

- To avoid electrical shock, remove all power from the inverter and wait at least five minutes prior to starting installation. Additionally, confirm that the DC bus voltage is less than 36 VDC.
- Installation should be performed only by qualified personnel.
- To avoid electrical shock, do not operate the inverter with the front cover or wiring cover removed, as accidental contact with exposed high-voltage terminals and internal components may occur.
- To prevent explosions or similar damage, ensure that all cables are properly connected to the correct terminals, and observe all wiring polarity indicators.

CAUTION

- Do not install or operate the interface if it is damaged or has parts missing.
- Prevent conductive items such as screws and metal fragments, or flammable substances such as oil, lint, paper fibers and sawdust from entering the inverter and interface card enclosure.
- Incorrect handling during installation or removal may cause equipment failure.
- Do not subject the cables to scratches, excessive stress, heavy loads or pinching.
- To prevent damage due to electrostatic discharge, always touch a grounded piece of metal prior to touching any equipment.
- Do not stand on or rest heavy objects on the equipment.
- To prevent burns from hot components, do not touch the inverter while power is on, or for some time after power is removed.
- Electrical noise may be emitted from the inverter, motor and wires. Always implement appropriate countermeasures to prevent nearby sensors and devices from malfunctioning due to such noise.

Operation

WARNING

- To avoid electrical shock, do not open the front cover of the inverter while power is on or while the inverter is running.
- To avoid electrical shock, do not operate switches with wet hands.
- If the inverter's function codes are incorrectly configured, or configured without adequate understanding of the appropriate inverter Operation Manual, the motor may rotate with a torque or at a speed not permitted for the machine. Confirm the settings of all function codes prior to running the inverter.

Maintenance, inspection, and parts replacement

WARNING

- To avoid electrical shock, remove all power from the inverter and wait at least five minutes prior to starting inspection. Additionally, confirm that the DC bus voltage is less than 36 VDC.
- Maintenance, inspection, and parts replacement should be performed only by qualified personnel.
- Remove all watches, rings and other metallic objects prior to starting work.
- To avoid electrical shock or other injuries, always use insulated tools.

Disposal

CAUTION

- Contact the local or state environmental agency in your area for details on the disposal of electrical components and packaging.

Other

WARNING

- Do not attempt to modify the equipment: doing so may cause electrical shock or injuries.
- For clarity purposes, illustrations in this manual may be drawn with covers or safety guards removed. Ensure all covers and safety guards are properly installed prior to starting operation.
- Do not perform hi-pot tests on the equipment.
- Performing a data initialization (set function code P00.18 to a value of 1) may reset all inverter function codes to their factory default settings. After performing this operation, remember to reenter any custom function code values prior to starting operation.

Icons

The following icons are used throughout this manual:



Indicates information which, if not heeded, can result in the product not operating to full efficiency, as well as information concerning incorrect operations and settings which may result in accidents.



Indicates information that can prove handy when performing certain settings or operations.



Indicates a reference to more detailed information.

– TABLE OF CONTENTS –

1	PRE-OPERATION INSTRUCTIONS.....	8
1.1	Product Overview	8
1.2	Features and Specifications	8
1.3	Unpacking and Product Confirmation	12
1.3.1	Shipment Confirmation	12
1.3.2	Component Overview	13
1.4	LED Indicators	14
1.4.1	Module Status LED	14
1.4.2	RS-485 Transmit (TX) LED	14
1.4.3	RS-485 Receive (RX) LED	14
2	INSTALLATION	15
2.1	Pre-Installation Instructions.....	15
2.2	Installation Procedure	15
3	WIRING.....	17
3.1	Terminals.....	17
3.2	Cabling	17
4	INVERTER FUNCTION CODE SETTINGS	18
5	FUNCTION CODE NUMBERING AND BEHAVIOR	20
5.1	Register Address	20
5.2	Function Code Synchronization	23
5.3	Internal Function Codes	24
5.3.1	Command Word (Function Code P100.01)	24
5.3.2	Motor Group Selection Word (Function Code P100.02)	25
5.3.3	Command Process Data (Function Codes P100.20 - P100.30)	25
5.3.4	Status Word (Function Code P100.51)	26
5.3.5	Motor Group Feedback Word (Function Code P100.52)	26
5.3.6	Operation Mode Word (Function Code P100.53)	27
5.3.7	Scanned Parameters Cycle Time (Function Code P100.54)	27
5.3.8	Number of Scanned Parameters (Function Code P100.55)	27
5.3.9	Max Invalid RX Count (Function Code P100.56)	27
5.3.10	Max Unexpected RX Count (Function Code P100.57)	27
5.3.11	Status Process Data (Function Codes P100.70 - P100.80)	27
5.3.12	General-Purpose Functions (Function Code Group P101)	30
5.3.13	Network Configuration Parameters (Function Codes P102.00 - P102.31)	30
5.3.14	Persistent User Parameters (Function Codes P102.32 - P102.63)	30
6	TIMEOUT PROCESSING	31
7	GALT ELECTRIC NETWORK PARAMETER UTILITY	32
7.1	Overview	32
7.2	Features	32
8	GALT ELECTRIC CONFIGURATION STUDIO	33
8.1	Overview	33
8.2	General Object Editing Activities.....	35

8.3	Device Settings	36
8.3.1	Status LED Settings.....	36
8.4	Process Data Configuration	37
8.4.1	Process Data Assignment Object.....	37
8.5	USB Virtual COM Port Settings.....	37
8.6	USB Serial Capture Window.....	38
8.7	Port Diagnostics	40
8.8	Batch Update Mode	40
8.9	Internal Logic Settings	41
8.9.1	Initial Persistent Values	41
8.9.2	Fail-safe Values	41
8.9.3	Database Logic.....	42
8.10	Manage Device Parameters.....	45
8.11	Monitor Device Parameters.....	46
8.12	Backup and Restore Parameters	47
8.13	Restore Factory Settings.....	48
8.14	Database.....	48
8.15	Help.....	48
9	FIRMWARE	49
9.1	Overview	49
9.2	Update Procedure.....	49
10	PROTOCOL-SPECIFIC INFORMATION.....	50
10.1	BACnet MS/TP.....	50
10.1.1	Protocol Implementation Conformance Statement (PICS)	50
10.1.2	Default Supported Objects.....	56
10.1.3	Server Settings	59
10.1.4	Node Settings	60
10.1.5	Device Object Settings	60
10.1.6	BACnet Object Settings.....	60
10.2	Modbus RTU	67
10.2.1	Overview	67
10.2.2	Holding & Input Registers.....	67
10.2.3	Coil & Discrete Input Mappings	67
10.2.4	Slave Settings.....	68
10.2.5	Node Settings	68
10.2.6	Holding/Input Register Remap Settings	68
10.3	Metasys N2.....	70
10.3.1	Default Supported Objects.....	71
10.3.2	Slave Settings.....	74
10.3.3	Node Settings	74
10.3.4	Metasys Object Settings.....	75
10.4	Siemens FLN (P1)	79
10.4.1	Default Supported Objects.....	80
10.4.2	Slave Settings.....	83
10.4.3	Node Settings	83
10.4.4	FLN Object Settings.....	84
10.5	DMX-512	87
10.5.1	Connections	87

10.5.2	Default Supported Objects.....	88
10.5.3	Slave Settings.....	88
10.5.4	Node Settings	88
10.5.5	DMX Parameter Settings	88
10.6	Generic Serial.....	89
10.6.1	Slave Settings.....	89
10.6.2	Transactions	89
10.6.3	Packet Data Objects	90
11	TROUBLESHOOTING	95

1 PRE-OPERATION INSTRUCTIONS

1.1 Product Overview

The G500XLT RS-485 multiprotocol communication interface allows information to be transferred seamlessly between a G500 series inverter and several different RS-485-based fieldbus networks with minimal configuration requirements. The interface installs directly onto the inverter's control board, and presents a pluggable, screw-style terminal block for connection to the RS-485 network.

Before using the interface, please familiarize yourself with the product and be sure to thoroughly read the instructions and precautions contained in this manual. In addition, please make sure that this instruction manual is delivered to the end user of the interface, and keep this instruction manual in a safe place for future reference or unit inspection.

Note that different interface firmware versions may provide varying levels of support for the various protocols. When using this manual, therefore, always keep in mind that the release date of the firmware version running on your interface as it must match this manual's respective release date in order for all documented aspects to apply.

The G500XLT may be referred to throughout the remainder of the manual as the device, interface, card, and option (or any combination thereof).

Supported Protocols

The interface currently provides support for the following fieldbus protocols:

- BACnet MS/TP Server
- Modbus RTU Slave
- Metasys N2 Slave
- Siemens FLN Slave
- DMX-512 Slave
- Generic Serial Slave

1.2 Features and Specifications

Table 1: Features

Item	Description
Multiple RS-485 Protocols	Active protocol and other network settings selected via software.
Galt Network Parameter Utility	Graphical user interface for configuring option card parameters for RS-485 networks.
Galt Electric Configuration Studio	Graphical user interface for discovery, object configuration, firmware update, and troubleshooting
Communication Loss Detection	Configurable actions for "fail-safe" conditions
PLC-Style Data Manipulation	Database logic allows construction of complex autonomous data conditioning
Field Upgradeable	Firmware updates automatically handled by the network parameter utility and configuration studio
Parameter Backup and Restore	Drive cloning

Table 2: General Hardware Specifications

Item	Description
Power Supply	Directly powered by the inverter or via USB
Grounding	Referenced to inverter's 5V power supply ground
LED Indicators	Module Status, RS-485 TX, RS-485 RX
USB Port	USB 2.0, mini-B 5-pin

Table 3: RS-485 Hardware Specifications

Item	Description
Number of Ports	1
Isolation	1.5kV galvanic isolation
Standard	TIA/EIA-485
Communication Speed and Duplex	300 - 250k baud (protocol dependent), half duplex
Connector Type	Pluggable, 4 position, screw-style terminal block
Cable Type	Twisted pair
Cable Length	Up to 1,200 m (4,000 ft)
Topologies	Daisy-chain
Unit Load	1/8

Table 4: BACnet MS/TP Specifications

Item	Description
Data Link	Master-Slave/Token Passing (MS/TP)
Protocol Revision	12
Standard Device Profile (Annex L)	BACnet Application Specific Controller (B-ASC)
BACnet Interoperability Building Blocks (BIBB)	ReadProperty-B (DS-RP-B), ReadPropertyMultiple-B (DS-RPM-B), WriteProperty-B (DS-WP-B), WritePropertyMultiple-B (DS-WPM-B), COV-B (DS-COV-B), Dynamic Device Binding-B (DM-DDB-B), Dynamic Object Binding-B (DM-DOB-B), DeviceCommunicationControl-B (DM-DCC-B), ReinitializeDevice-B (DM-RD-B)
Segmentation	Not supported
Max APDU Length	480 bytes
Character Sets	ISO 10646 (UTF-8)
Object Types	Analog Output, Analog Input, Analog Value, Binary Output, Binary Input, Binary Value, Device, Multi-state Output, Multi-state Input, Multi-state Value
Priority Array	Yes
Baud Rates	9600, 19200, 38400, 57600, 76800, 115200

Item	Description
Parity	No Parity
Stop Bits	1
Data Bits	8

Table 5: Modbus RTU Specifications

Item	Description
Read Function Codes	Read coils (1), Read input status (2), Read multiple registers (3), Read input registers (4), Diagnostics (8)
Write Function Codes	Write coil (5), Write single register (6), Force multiple coils (15), Write multiple registers (16)
Max Read Register Size	125 registers
Max Write Register Size	123 registers
Register Data Type	16-bit integer
Baud Rates	2400, 4800, 9600, 19200, 38400, 57600, 115200
Parity	No Parity, Odd Parity, Even Parity
Stop Bits	1, 2
Data Bits	8

Table 6: Metasys N2 Specifications

Item	Description
Object Types	Analog Output, Analog Input, Binary Output, Binary Input, Internal Float (ADF), Internal Integer (ADI), Internal Byte (BD)
Baud Rates	9600
Parity	No Parity
Stop Bits	1
Data Bits	8

Table 7: Siemens FLN Specifications

Item	Description
Application Number	18260
Application Descriptor	GALT G500
Revision String	MP10
Revision Number	1
Object Types	LAI, LAO, LDI, LDO
Baud Rates	2400, 4800, 9600, 19200, 38400, 57600, 115200
Parity	No Parity
Stop Bits	1
Data Bits	8

Table 8: DMX-512 Specifications

Item	Description
RDM Support	No
Baud Rates	250000
Parity	No Parity
Stop Bits	2
Data Bits	8

Table 9: Generic Serial Specifications

Item	Description
Packet Definition	Transaction-based
Packet Delimiter	Packet Gap Interval (Character Times)
Encoding	Binary, ASCII
Error Checking	Checksum, CRC
Baud Rates	300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
Parity	No Parity, Odd Parity, Even Parity
Stop Bits	1, 2
Data Bits	7, 8

Table 10: Applicable Inverters

Series	Type	Capacity	Software version
G500	G5□□-□□□□□UL-□□	All capacities	All versions

Table 11: Environmental Specifications

Item	Description
Environmental Specifications	Refer to the G500 Operation Manual.

1.3 Unpacking and Product Confirmation

1.3.1 Shipment Confirmation

Check the enclosed items. Confirm that the correct quantity of each item was received, and that no damage occurred during shipment.

- G500XLT interface board (refer to Figure 1).
- One separate M3 x 6mm mounting screw (see Figure 2).
- One USB mini-B cable (see Figure 3).

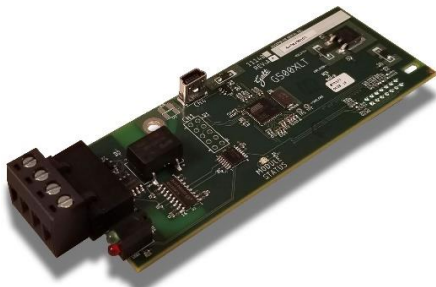


Figure 1: G500XLT Interface Board



Figure 2: M3 x 6mm Mounting Screw



Figure 3: USB Mini-B Cable

1.3.2 Component Overview

Figure 4 provides an overview of the important interface card components.

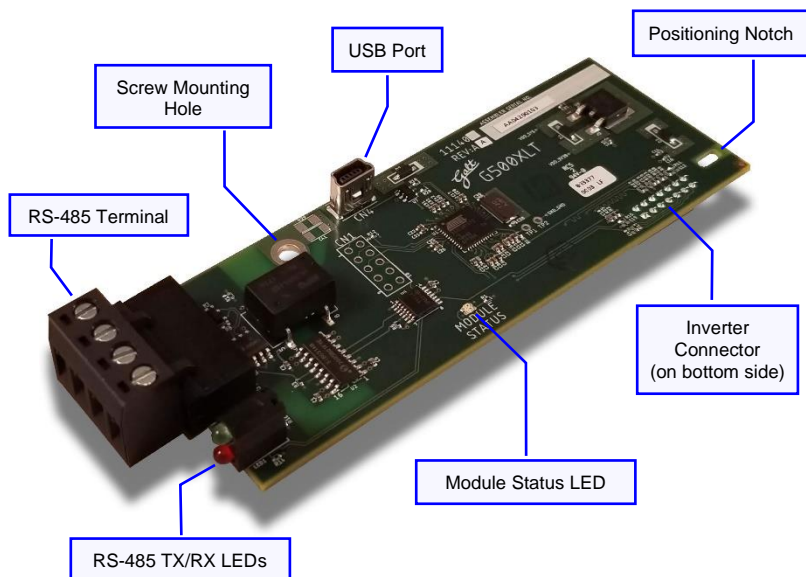


Figure 4: G500XLT Component Overview

Positioning Notch

Aligns with the positioning key on the inverter chassis to ensure that the interface card is installed into the correct communication port (refer to section 2.2).

RS-485 Terminal

A pluggable, 4-position terminal block is provided for connection to an RS-485 network. Wiring must be performed in a daisy-chain fashion between all devices on the network. This port is galvanically isolated from all other circuitry on the card.

Standoff Mounting Hardware

The provided M3 x 6mm screw is used to secure the card to the standoff located on the inverter's control board. Refer to section 2.2.

Inverter Control Board Connector

Attaches to the inverter's connector board, which may vary depending on the inverter model.

USB Port

USB 2.0 port with mini-B connector. Used to access the card via the Galt Electric Configuration Studio (refer to section 6). The USB connection can supply power to the card when the card is not installed on an inverter or the inverter is powered off.

Module Status LED

This LED indicates the current status of the interface card. Refer to section 1.4.

RS-485 TX/RX LEDs

A pair of LEDs are provided on the RS-485 port for transmit and receive activity. Refer to section 1.4.

1.4 LED Indicators

1.4.1 Module Status LED

LED Activity	Status	Note
Off	Device Off	Power is not applied to the option card
Green Blink, Red Blink	Startup	Startup blink sequence
Green Blink (Rapid)	Sync	The option card is synchronizing parameters with the inverter
Green On	Device On	Normal status
Green Blink	USB Connection	An active USB connection is present
Red Blink	Error Code	Record the error code sequence and refer to section 11 for troubleshooting information

1.4.2 RS-485 Transmit (TX) LED

LED Activity	Status	Note
Off	Not Transmitting	The device is not transmitting data on the RS-485 network
Green Blink	Transmitting	The device is transmitting data on the RS-485 network
Green On	Transmitting	The device is transmitting data rapidly on the RS-485 network

1.4.3 RS-485 Receive (RX) LED

LED Activity	Status	Note
Off	Not Receiving	The device is not receiving data on the RS-485 network
Green Blink	Receiving	The device is receiving data on the RS-485 network
Green On	Receiving	The device is receiving data rapidly on the RS-485 network

2 INSTALLATION

2.1 Pre-Installation Instructions

WARNING

- To avoid electrical shock, remove all power from the inverter and wait at least five minutes prior to starting installation. Additionally, confirm that the DC bus voltage is less than 36 VDC.
- Installation should be performed only by qualified personnel.
- To avoid electrical shock, do not operate the inverter with the front cover or wiring cover removed, as accidental contact with exposed high-voltage terminals and internal components may occur.
- To prevent explosions or similar damage, ensure that all cables are properly connected to the correct terminals, and observe all wiring polarity indicators.
- Additional option cards may be used when the G500XLT is installed in the inverter. Please consult with the factory first to confirm compatibility.

2.2 Installation Procedure

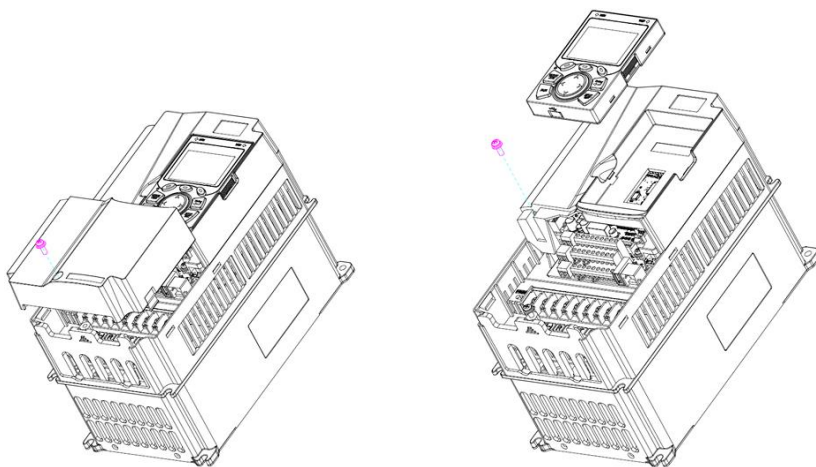


Before installing the interface card, perform all wiring for the main circuit terminals and control circuit terminals.

1. Remove the front cover from the inverter to expose the control printed circuit board (control PCB).



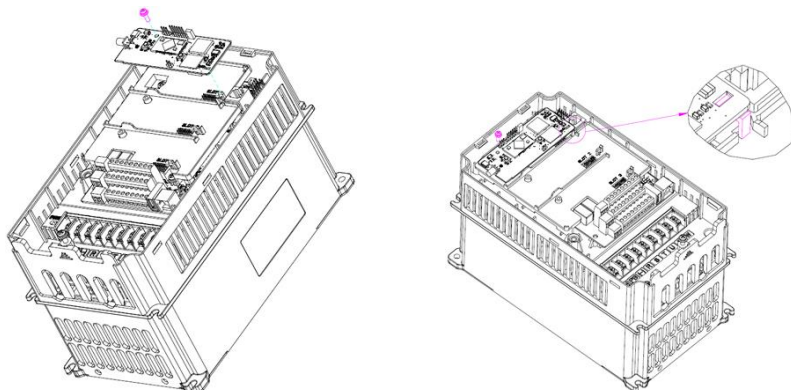
For installation instructions, refer to the G500 Operation Manual.



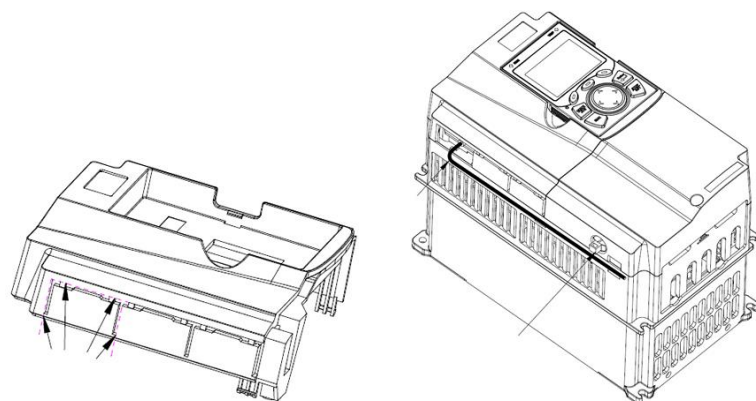
2. Engage connector CN3 (on the back of the interface card) into the "Slot 1", "Slot 2", or "Slot 3" connector on the connector board. Ensure that the connectors are fully engaged.



Ensure that the interface card is fully aligned and seated into the communication port. Failure to do so may lead to insufficient connector insertion and result in contact failure.



3. Secure the interface card to the connector board PCB by tightening the included M3 x 6mm screw into the standoff mounting hole.
4. Cut off the appropriate tabs on the side of the inverter's front cover and open the knockout for cabling.
5. Connect the network cable(s) as necessary. Wire the RS-485 cable(s) to the RS-485 terminal block, making sure that all wires are fully seated. Ensure that the cables are routed in such a way that they will not be pinched and are not located near any power-carrying wiring, such as the inverter's input power or motor wires.



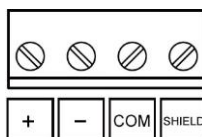
6. Reinstall all covers removed in step 1. Take a moment to confirm that the RS-485 cable(s) are not being pinched and are not routed near any power-carrying wiring.



For reinstallation instructions, refer to the G500 Operation Manual.

3 WIRING

3.1 Terminals



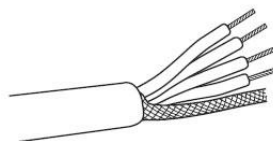
Terminal	Network Connection
+	RS-485 Positive (Non-Inverting) Data Signal
-	RS-485 Negative (Inverting) Data Signal
COM	RS-485 Common-Mode (0V) Reference
SHIELD	Cable Shielding Daisy-Chain Point This terminal has no internal connection

Be sure to check the following before connecting the inverter to the network.

- Check that the G500XLT is securely inserted into the inverter. (Refer to section 2.2.)
- Check that the correct network settings are set for the option board. (Refer to section 6.)
- Check that all RS-485 wires are firmly connected to the G500XLT.

3.2 Cabling

Use network connection cable rated for use with RS-485 networks. Because 3 signals are required (+, -, and COM), it is recommended to use either a 1.5 twisted pair or 2 twisted pair cable. A shielded cable is recommended. The cable shield should be connected to the SHIELD terminal of each card on the network and terminated to earth ground at one end only. All RS-485 devices must be wired in a daisy-chain topology. The RS-485 cabling must not exceed 4,000ft (~1200m). If the RS-485 network requires a distance greater than this limit, use RS-485 repeaters to divide the network into segments, where each segment does not exceed 4,000ft (~1200m).



End-of-line termination resistors should only be installed when necessary, according to the cable length and baud rate used. The termination resistors must match the characteristic impedance of the cable (typically 120 ohms) and should be installed at either end of the RS-485 network.

Table 12: Termination Baud Rates and Distances

Baud Rate	Maximum Length Without Termination
9600 and below	Termination not required
19,200	2,800ft (850m)
38,400	1,400ft (425m)
57,600	940ft (285m)
76,800	700ft (210m)
115,200	470ft (140m)

Recommended Cable Characteristics

- 1 twisted-pair for RS-485 + and - data signals and 1 conductor (or twisted-pair) for RS-485 common-mode reference (COM)
- Wire gauge of 24 AWG to 16 AWG
- Characteristic impedance of 100Ω to 140Ω
- Capacitance (wire to wire) of less than 30pF/ft (~100pF/m)
- Capacitance (wire to shield) of less than 60pF/ft (~200pF/m)

4 INVERTER FUNCTION CODE SETTINGS

The inverter function codes listed in Table 13 and Table 14 are important for proper operation of the end-to-end communication system. Some of these parameters must be set to specific values, and some may have multiple allowable settings depending on the desired operation of the overall application. Although there may be many other function codes that will require configuration for your specific application, it is important to understand the manner in which the following function codes will impact successful communications with, and control of, the inverter.



For further details regarding these function codes, please refer to the **G500 Operation Manual**.

Table 13: Inverter Control Function Codes

Function Code	Name	Detailed Parameter Description	Required Value for Control via G500XLT
P00.01	Running command channel	0: Keypad 1: Terminal 2: Communication	2: Communication
P00.02	Communication running command channel	0: Modbus 1: PROFIBUS/CANopen/DeviceNET/G500XLT 2: Ethernet 3: EtherCat/Profinet/EtherNet/IP/G500ETH 4: PLC programmable card 5: Wireless communication card Note: 1, 2, 3, 4 and 5 are extended functions which are applicable with corresponding cards.	1: PROFIBUS/CANopen/DeviceNET/G500XLT
P00.06	A frequency command selection	0: Set via keypad 1: Set via AI1 2: Set via AI2 3: Set via AI3 4: Set via high speed pulse HDIA 5: Set via simple PLC program 6: Set via multi-step speed running 7: Set via PIDcontrol 8: Set via Modbus communication 9: Set via PROFIBUS/CANopen/DeviceNET/G500XLT communication 10: Set via Ethernet communication 11: Set via high speed pulse HDIB 12: Set via pulse string AB 13: Set via EtherCat/Profinet/EtherNet/IP/G500ETH communication 14: Set via PLC card 15: Reserved	9: Set via PROFIBUS/CANopen/DeviceNET/G500XLT communication
P00.07	B frequency command selection		
P00.09	Combination mode of settings source	0: A 1: B 2: (A + B) 3: (A - B) 4: Max(A, B) 5: Min(A, B)	0: A 1: B 2: (A + B) 3: (A - B) 4: Max(A, B) 5: Min(A, B)

Table 14: Inverter Communication Error Function Codes

Function Code	Name	Detailed Parameter Description	Associated Fault Code
P16.24	Slot 1 extension card identification time	0.0 - 600.0s (Default value 0.0s)	60: Card slot 1 identification failure (F1-Er)
P16.25	Slot 2 extension card identification time	Sets the time allowed for the extension card to identify itself after the inverter has detected the card.	61: Card slot 2 identification failure (F2-Er)
P16.26	Slot 3 extension card identification time	When this parameter is set to 0.0, identification fault detection is not performed.	62: Card slot 3 identification failure (F3-Er)
P16.27	Slot 1 extension card communication timeout time	0.0 - 600.0s (Default value 5.0s)	63: Card slot 1 card communication timeout fault (C1-Er)
P16.28	Slot 2 extension card communication timeout time	Sets the maximum amount of time that can elapse without the inverter receiving communications from the extension card after the inverter has detected the card.	64: Card slot 2 card communication timeout fault (C2-Er)
P16.29	Slot 3 extension card communication timeout time	When this parameter is set to 0.0, card communication timeout fault detection is not performed. It is recommended that this parameter be kept at its default value of 5.0s.	65: Card slot 3 card communication timeout fault (C3-Er)
P15.43	G500XLT card timeout time	0.0 - 600.0s (Default value 0.0s) Sets the amount of time that a network timeout event must be active to cause the inverter to fault. If network communications recover before this time has expired, no fault is triggered. When this parameter is set to 0.0, network timeout faults are not triggered. Refer to section 6 for details on this feature.	72: G500XLT fault (E-E72)

5 FUNCTION CODE NUMBERING AND BEHAVIOR

5.1 Register Address

All accessible inverter function codes can be referenced by their register address and can be conveniently referenced in the configuration studio (section 8.10). These same register addresses are used when accessing function codes via certain RS-485 protocols. The terms “function code” and “register” refer to data stored on the inverter and will be used interchangeably throughout this documentation. For a complete list of function codes, refer to the **G500 Series Inverter Operation Manual**.

All inverter function codes are exposed as register addresses according to a mathematical conversion formula which combines two elements (the function code group and the function code number) to create a unique register address for each function code. To determine the register address for a given function code, therefore, the function code group is first multiplied by 100, then added to the function code number. This operation is expressed mathematically via Equation 1. The function code to register address conversion examples are shown in the Table 15.

$$\text{Register Address} = (\text{Function Code Group} \times 100) + \text{Function Code Number}$$

Equation 1

For convenience, the function code to register address mapping can be referenced in the Configuration Studio (section 8.10).

Note that not all of the available registers that exist in the interface card’s register map have corresponding function codes that exist in the inverter. In other words, if a read from or write to a register address that does not correspond to an existing inverter function code takes place, the read/write may be successful (depending on the specific register accessed; refer to section 5.2), but the data will have no meaning. This feature is beneficial in situations where the accessing of non-contiguous registers can be made more efficient by accessing an all-inclusive block of registers (some of which correspond to inverter function codes and some of which do not), while only manipulating those in your local programming that are known to exist.

Table 15: Function Code-to-Register Address Conversion Examples

Function Code Group	Group Number	Register Address Example Using Equation 1
P00: Basic functions	0	P00.00 (speed control mode): $(0 \times 100) + 0 = 0$: P00.99: $(0 \times 100) + 99 = 99$
P01: Start/stop control	1	P01.00 (running mode of start): $(1 \times 100) + 0 = 100$: P01.99: $(1 \times 100) + 99 = 199$
P02: Parameters of motor 1	2	P02.00 (type of motor 1): $(2 \times 100) + 0 = 200$: P02.99: $(2 \times 100) + 99 = 299$
P03: Vector control of motor 1	3	P03.00 (speed loop proportional gain 1): $(3 \times 100) + 0 = 300$: P03.99: $(3 \times 100) + 99 = 399$
P04: V/F control	4	P04.00 (V/F curve setup of motor 1): $(4 \times 100) + 0 = 400$: P04.99 $(4 \times 100) + 99 = 499$
P05: Input terminals	5	P05.00 (HDI input type): $(5 \times 100) + 0 = 500$: P05.99: $(5 \times 100) + 99 = 599$
P06: Output terminals	6	P06.00 (HDO output type): $(6 \times 100) + 0 = 600$: P06.99: $(6 \times 100) + 99 = 699$

Function Code Group	Group Number	Register Address Example Using Equation 1
P07: HMI	7	P07.00 (User password): $(7 \times 100) + 0 = 700$: P07.99: $(7 \times 100) + 99 = 799$
P08: Enhanced functions	8	P08.00 (Acceleration time 2): $(8 \times 100) + 0 = 800$: P08.99: $(8 \times 100) + 9 = 899$
P09: PID control	9	P09.00 (PID reference source): $(9 \times 100) + 0 = 900$: P09.99: $(9 \times 100) + 99 = 999$
P10: Simple PLC and multi-step speed control	10	P10.00 (Simple PLC mode): $(10 \times 100) + 0 = 1000$: P10.99: $(10 \times 100) + 99 = 1099$
P11: Protection parameters	11	P11.00 (Phase-loss protection): $(11 \times 100) + 0 = 1100$: P11.99: $(11 \times 100) + 99 = 1199$
P12: Parameters of motor 2	12	P12.00 (Type of motor 2): $(12 \times 100) + 0 = 1200$: P12.99: $(12 \times 100) + 99 = 1299$
P13: Control parameters of synchronous motor	13	P13.00 (Reduction rate of the injection current of synchronous motor): $(13 \times 100) + 0 = 1300$: P13.99: $(13 \times 100) + 99 = 1399$
P14: Serial communication function	14	P14.00 (Local communication address): $(14 \times 100) + 0 = 1400$: P14.99: $(14 \times 100) + 99 = 1499$
P15: Functions of communication extension card 1	15	P15.00 (Extension Card 1 Type): $(15 \times 100) + 0 = 1500$: P15.99: $(15 \times 100) + 99 = 1599$
P16: Functions of communication extension card 2	16	P16.00 (Extension Card 2 Type): $(16 \times 100) + 0 = 1600$: P16.99: $(16 \times 100) + 99 = 1699$
P17: State-check functions	17	P17.00 (Set frequency): $(17 \times 100) + 0 = 1700$: P17.99: $(17 \times 100) + 99 = 1799$
P18: Closed-loop control state check	18	P18.00 (Actual frequency of encoder): $(18 \times 100) + 0 = 1800$: P18.99: $(18 \times 100) + 99 = 1899$
P19: Extension card state check	19	P19.00 (State of card slot 1): $(19 \times 100) + 0 = 1900$: P19.99: $(19 \times 100) + 99 = 1999$
P20: Encoder of motor 1	20	P20.00 (Encoder type display): $(20 \times 100) + 0 = 2000$: P20.99: $(20 \times 100) + 99 = 2099$
P21: Position control	21	P21.00 (Positioning mode): $(21 \times 100) + 0 = 2100$: P21.99: $(21 \times 100) + 99 = 2199$
P22: Spindle positioning	22	P22.00 (Spindle positioning mode selection): $(22 \times 100) + 0 = 2200$: P22.99: $(22 \times 100) + 99 = 2299$
P23: Vector control of motor 2	23	P23.00 (Speed loop proportional gain 1): $(23 \times 100) + 0 = 2300$: P23.99: $(23 \times 100) + 99 = 2399$

Function Code Group	Group Number	Register Address Example Using Equation 1
P24: Encoder of motor 2	24	P24.00 (Encoder type display): $(24 \times 100) + 0 = 2400$: P24.99: $(24 \times 100) + 99 = 2499$
P25: Extension I/O card input functions	25	P25.00 (HDI3 input type selection): $(25 \times 100) + 0 = 2500$: P25.99: $(39 \times 100) + 99 = 2599$
P26: Output functions of extension I/O card	26	P26.00 (HDO2 output type): $(26 \times 100) + 0 = 2600$: P26.99: $(26 \times 100) + 99 = 2699$
P27: PLC functions	27	P27.00 (PLC function enable): $(27 \times 100) + 0 = 2700$: P27.99: $(27 \times 100) + 99 = 2799$
P28: Master/slave control functions	28	P28.00 (Master/slave mode selection): $(28 \times 100) + 0 = 2800$: P28.99: $(28 \times 100) + 99 = 2899$
P90: Customized function group 1	90	P90.00: $(90 \times 100) + 0 = 9000$: P90.99: $(90 \times 100) + 99 = 9099$
P91: Customized function group 2	91	P91.00: $(91 \times 100) + 0 = 9100$: P91.99: $(91 \times 100) + 99 = 9199$
P92: Customized function group 3	92	P92.00: $(92 \times 100) + 0 = 9200$: P92.99: $(92 \times 100) + 99 = 9299$
P93: Customized function group 4	93	P93.00: $(93 \times 100) + 0 = 9300$: P93.99: $(93 \times 100) + 99 = 9399$
P100: Control/status functions	100	P100.00: $(100 \times 100) + 0 = 10000$: P100.99: $(100 \times 100) + 99 = 10099$
P101: General- purpose functions	101	P101.00: $(101 \times 100) + 0 = 10100$: P101.99: $(101 \times 100) + 99 = 10199$
P102: Network configuration/persistent user parameters	102	P102.00: $(102 \times 100) + 0 = 10200$: P102.63: $(102 \times 100) + 63 = 10263$

5.2 Function Code Synchronization

The option card uses an internal database to cache all of the inverter's function codes. These function codes are constantly being read and/or written (as applicable), and their current values are therefore mirrored in the interface card's internal memory. The inverter provides two methods to synchronize parameters with the option card.

A limited number of parameters can be synchronized using high-speed, cyclic access. This method provides very fast function code synchronization by using a block of command process data and a block of status process data that is exchanged between the inverter and option card every communication cycle. Refer to sections 5.3.3 and 5.3.11 for more details on process data parameters.

The other method of synchronization is performed on a request-response basis, providing a single read or write transaction per communication cycle. Synchronization of these function codes is much slower than process data parameters and depends on the total number of function codes that exist on the inverter. Parameters that are synchronized using this method are referred to as "scanned" parameters. The card automatically detects the function codes that exist on the inverter at startup using this synchronization method.

This synchronization approach provides quick and deterministic response times to network requests and also allows the option card's network driver to be decoupled from inverter communications. Because of this decoupling, accesses to any function code (Pxx.00 to Pxx.99, where "xx" is any valid function code group from Table 15) will always be successful. Even if an inverter function code corresponding to a given register does not exist, the interface card still maintains a placeholder location in its internal mirroring memory for that register. This feature allows for the block access of non-contiguous registers, as described in section 5.1.

One of the principle disadvantages of parameter caching is that write data checking is not available. This means that when the value of a parameter is modified via a network protocol, the interface card itself is not able to determine if the new value will be accepted by the inverter (the value may be out-of-range, or the inverter may be in a state in which it will not accept new values being written via communications, etc.) For example, if a write is performed to a command function code with a data value that is out-of-range, the interface card will not generate a corresponding error. However, the register can be read over the network at a later time to confirm whether or not that the written value "took hold" in the inverter.

5.3 Internal Function Codes

The interface card defines special internal parameters that do not exist on the inverter and are only available in the card's internal memory. These parameters provide command and status information that are otherwise unavailable on the inverter.

5.3.1 Command Word (Function Code P100.01)

The command word bit-mapping is described in Table 16.

Table 16: Command Word Bit-Mapping

Bit	Name	Value	Description
0	Start/Stop ¹	1	Start the drive
		0	Stop the drive
1	Reverse/Forward	1	Reverse
		0	Forward
2	Jog Operation	1	Jog enabled
		0	Jog disabled
3	Coast Stop ¹	1	Coast to stop
		0	No effect
4	Emergency Stop ¹	1	Emergency stop
		0	No effect
5	Reset Fault ^{1,2}	1	Reset fault
		0	No effect
6...10	Reserved		
11	Control Mode Switching	1	Switch torque/speed control mode
		0	Use current torque/speed control mode
12	Clear Power Consumption	1	Clear power consumption
		0	No effect
13	Pre-excitation Command	1	Enable pre-excitation
		0	Pre-excitation disabled
14	DC Brake Command	1	Enable DC brake
		0	DC brake disabled
15	Heartbeat Command	1	Set heartbeat value to 1
		0	Set heartbeat value to 0

1. These command bits are evaluated by the card using a priority, from highest to lowest, as follows: Reset Fault, Emergency Stop, Coast Stop, Start/Stop. Lower priority bits are automatically cleared by the card when higher priority bits are set. Higher priority bits must be cleared before lower priority bits may be set.
2. This command bit is ignored if the drive is not faulted.

5.3.2 Motor Group Selection Word (Function Code P100.02)

The motor group selection word is described in Table 17.

Table 17: Motor Group Selection Word

Name	Value	Description
Motor Group Selection	0	Select motor group 1
	1	Select motor group 2
	2...65535	Reserved

5.3.3 Command Process Data (Function Codes P100.20 - P100.30)

These parameters are sent to the inverter from the card at high speed, every communication cycle and are used to command the inverter. There are 11 parameters available (Command Process Data 0 - 10). Each parameter is a placeholder for a single process data item. Section 8.4 provides details on configuring the data assigned to these parameters.

Table 18 details the default command process data. Since the process data mapping is configurable, the system integrator is responsible for managing, maintaining, and documenting the actual configuration. Always use the studio to confirm the configuration before commissioning the device.

Table 18: Default Command Process Data Mapping

Command Parameter	Process Data Assignment	Description
P100.20: Command Process Data 0	Frequency command	0.01Hz
P100.21: Command Process Data 1	PID reference	0.1%
P100.22: Command Process Data 2	PID feedback	0.1%
P100.23: Command Process Data 3	Torque command	0.1%
P100.24: Command Process Data 4	Voltage command	0.1%

5.3.4 Status Word (Function Code P100.51)

The status word bit-mapping is described in Table 19.

Table 19: Status Word Bit-Mapping

Bit	Name	Value	Description
0	Running/Stopped	1	Inverter is running
		0	Inverter is stopped
1	Reverse/Forward ¹	1	Reverse
		0	Forward
2	Inverter Fault	1	Faulted
		0	Not faulted
3	POFF	1	Inverter in POFF state
		0	Inverter not in POFF state
4...7	Reserved		
8	Bus Voltage Established	1	Ready to run
		0	Not ready to run
9...10	Reserved		
11	Motor Type Feedback	1	Synchronous motor
		0	Asynchronous motor
12	Overload Warning	1	Overload pre-alarm
		0	No overload pre-alarm
13...14	Reserved		
15	Heartbeat Feedback	1	Heartbeat value is 1
		0	Heartbeat value is 0

1. Only updated when Running/Stopped bit is 1, otherwise shows last running direction.

5.3.5 Motor Group Feedback Word (Function Code P100.52)

The motor group feedback word is described in Table 20.

Table 20: Motor Group Feedback Word

Name	Value	Description
Motor Group Feedback	0	Motor group 1 selected
	1	Motor group 2 selected
	2...65535	Reserved

5.3.6 Operation Mode Word (Function Code P100.53)

The operation mode word is described in Table 21.

Table 21: Operation Mode Word

Name	Value	Description
Run/Stop Mode	0	Keyboard control
	1	Terminal control
	2	Communication control
	3...65535	Reserved

5.3.7 Scanned Parameters Cycle Time (Function Code P100.54)

The measurement, in milliseconds, of the last parameter scan cycle time. This is the amount of time it took the option card to synchronize all scanned parameters with the inverter during the last parameter scan cycle.

5.3.8 Number of Scanned Parameters (Function Code P100.55)

This parameter shows the number of parameters that were synchronized during the last parameter scan cycle. Because the option card uses an optimization algorithm to automatically detect the parameters that exist on the inverter, the number of parameters for the first scan cycle will be the maximum number of possible parameters. For all subsequent scan cycles, this value will reflect the actual number of parameters that exist on the inverter and are being synchronized.

5.3.9 Max Invalid RX Count (Function Code P100.56)

The maximum number of consecutive, invalid frames received by the option card from the inverter. This parameter is a metric of the communication health between the option card and inverter. The value may be reset by writing a 0 to the parameter so that a new maximum can be determined.

5.3.10 Max Unexpected RX Count (Function Code P100.57)

The maximum number of consecutive, valid, but unexpected frames received by the option card from the inverter. This parameter is a metric of the communication health between the option card and inverter. The value may be reset by writing a 0 to the parameter so that a new maximum can be determined.

5.3.11 Status Process Data (Function Codes P100.70 - P100.80)

These parameters are received by the card from the inverter at high speed, every communication cycle and are used to monitor the status of the inverter. There are 11 parameters available (Status Process

Data 0 - 10). Each parameter is a placeholder for a single process data item. Section 8.4 provides details on configuring the data assigned to these parameters.

Table 22 details the default status process data. Since the process data mapping is configurable, the system integrator is responsible for managing, maintaining, and documenting the actual configuration. Always use the studio to confirm the configuration before commissioning the device.

Table 22: Default Status Process Data Mapping

Status Parameter	Process Data Assignment	Description
P100.70: Status Process Data 0	Status word 2	See Table 23
P100.71: Status Process Data 1	Running frequency	0.01Hz
P100.72: Status Process Data 2	Frequency reference	0.01Hz
P100.73: Status Process Data 3	Bus voltage	0.1V
P100.74: Status Process Data 4	Output voltage	V
P100.75: Status Process Data 5	Output current	0.1A
P100.76: Status Process Data 6	Actual output torque	0.1%
P100.77: Status Process Data 7	Actual output power	0.1%
P100.78: Status Process Data 8	Fault code	Refer to P07.27 description in the G500 Operation Manual
P100.79: Status Process Data 9	PID reference	0.1%
P100.80: Status Process Data 10	PID feedback	0.1%

Table 23: Status Word 2 Bit-Mapping

Bit	Name	Value	Description
0	Jogging	1	Inverter is jogging
		0	Inverter is not jogging
1	Frequency Level Detection FDT1	1	Output frequency has exceeded the FDT1 detection value
		0	Output frequency has decreased below the FDT1 detection value
2	Frequency Level Detection FDT2	1	Output frequency has exceeded the FDT2 detection value
		0	Output frequency has decreased below the FDT2 detection value
3	Frequency Arrival	1	Output frequency is within the detection range of the set frequency
		0	Output frequency is outside of the detection range of the set frequency
4	Running at Zero Speed	1	Inverter is running at zero speed
		0	Inverter in not running at zero speed
5	Upper Frequency Reached	1	The inverter is running at the upper limit frequency
		0	The inverter is not running at the upper limit frequency
6	Lower Frequency Reached	1	The inverter is running at the lower limit frequency
		0	The inverter is not running at the lower limit frequency
7	Pre-excitation Feedback	1	Pre-excitation enabled
		0	Pre-excitation disabled
8	Underload Warning	1	Underload pre-alarm
		0	No underload pre-alarm
9	Simple PLC State Completed	1	The simple PLC's current stage has completed
		0	The simple PLC's current stage has not completed
10	Simple PLC Cycle Completed	1	The simple PLC's cycle has completed
		0	The simple PLC's cycle has not completed
11	Positioning Completed	1	Position control positioning completed
		0	Position control positioning not completed
12	Spindle Zeroing Completed	1	Spindle zeroing completed
		0	Spindle zeroing not completed
13	Spindle Scale-Division Completed	1	Spindle scale-division completed
		0	Spindle scale-division not completed
14	Speed Limiting	1	The speed has been limited while in torque control
		0	The speed has not been limited
15	Speed/Position Control Switching Completed	1	Speed/position control switch-over completed
		0	Speed/position control switch-over not completed

5.3.12 General-Purpose Functions (Function Code Group P101)

The parameters in this group serve as free parameters that can be used for any purpose. For example, these parameters can be used as “scratch-pad” memory for intermediate database logic operations (see section 8.9.3 for details on database logic).

5.3.13 Network Configuration Parameters (Function Codes P102.00 - P102.31)

These parameters allow for dynamic configuration of the card's RS-485 port at run time. These parameters are used by the Network Parameter Utility (see section 6) and are not intended to be accessed or modified by other means.

5.3.14 Persistent User Parameters (Function Codes P102.32 - P102.63)

These parameters provide general-purpose, persistent memory. This is useful for parameters that must retain their values across power cycles and reboots. These parameters are initialized by defining an initializer object using the Configuration Studio (refer to section 8.9.1 for details).



Care must be taken when using the persistent user parameters, because this memory is backed by internal flash storage and is susceptible to the write cycle limitations of flash technology. Therefore, it is important to only use these parameters for data which changes infrequently.

6 TIMEOUT PROCESSING

The interface card can detect breaks in network communications and trigger a timeout event. The card's timeout time is configured on a per-protocol basis by the timeout time setting. Refer to the specific protocol section in section 10 for details on this setting. If this setting is greater than 0, a timeout event will be triggered when a break in network communications exceeds the configured timeout time. If the timeout time is set to 0, timeout processing will be disabled.

The card supports two, simultaneous timeout actions when a timeout event is triggered. When timeout processing is enabled, the card continuously reports the health of the network to the inverter using a timeout flag. The timeout flag is set when a timeout event is triggered. The timeout flag is cleared when network communications resume. The inverter can be configured to fault if the timeout flag remains set for a specific period of time. Refer to P15.43 in section 4 for details on this inverter setting.

In addition to reporting the health of the network to the inverter using the timeout flag, the card can also write fail-safe values to any inverter function codes when timeout processing is enabled and a timeout event is triggered. Refer to section 8.9.2 for details on how to configure fail-safe values.

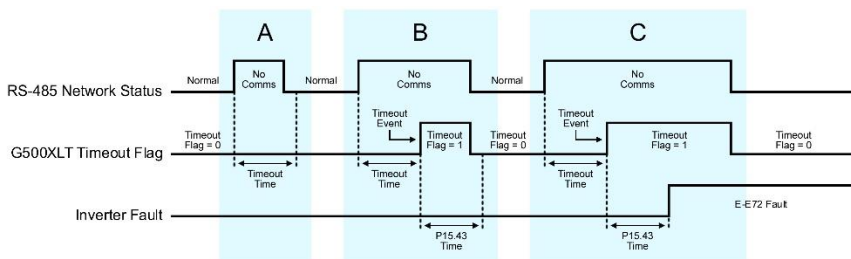


Figure 5: Timeout Processing Diagram

Figure 5 above shows the relative timing of the various signals, settings, and events associated with timeout processing. The diagram depicts three different scenarios:

- RS-485 communications are briefly interrupted, but recover before the card's timeout time has elapsed. Therefore, no timeout event is triggered and the card's timeout flag remains deasserted.
- RS-485 communications are interrupted for a period of time exceeding the card's timeout time. This triggers a timeout event and the card asserts its timeout flag. However, communications recover and the card deasserts its timeout flag before the inverter's P15.43 time has elapsed. Therefore, no fault is triggered.
- RS-485 communications are interrupted for a period of time exceeding the card's timeout time. This triggers a timeout event and the card asserts its timeout flag. The timeout flag remains asserted for a period of time exceeding the inverter's P15.43 time and the inverter faults. Communications recover and the card deasserts its timeout flag. However, the inverter remains faulted until the fault is reset.

7 GALT ELECTRIC NETWORK PARAMETER UTILITY

7.1 Overview

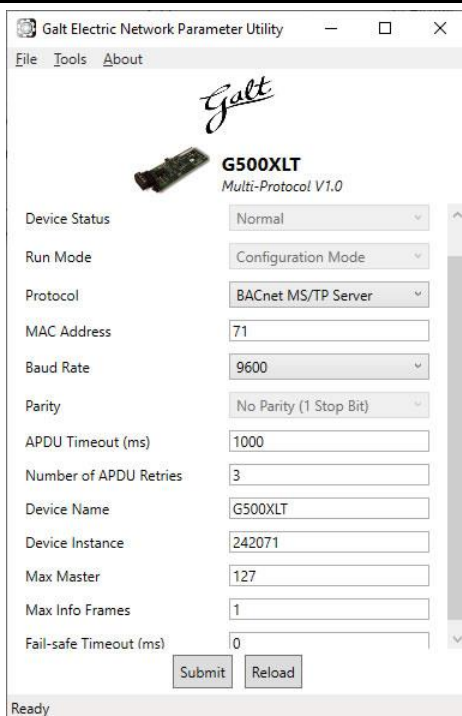


Figure 6: Galt Electric Network Parameter Utility

The Network Parameter Utility (NPU) software is an end-user application intended to quickly and easily configure the option card's network settings in the field via USB. The communication settings for the RS-485 port on the option card are dependent upon the system in which the inverter is being installed. The NPU is designed to configure only these network communication settings. The latest release of the NPU can be downloaded from the [product web page](#).

7.2 Features

- Light-weight, portable application - no installation necessary, simply unzip and run the exe.
- Includes all necessary USB drivers and firmware files.
- Automatically discovers and connects to the device via USB.
- Allows configuration of standard, field-configurable network settings such as protocol, baud rate, parity, address, and other protocol-specific settings.
- Detects outdated firmware on the connected device and allows the user to update the firmware on the device to the latest version.
- Only shows those protocols and options actually configured on the connected device.
- Self-documented - each setting provides a useful tooltip to the user that describes the setting.



Note

The NPU software writes to the card's network configuration parameters (P102.00 - P102.31). Refer to section 5.3.13 for details.

8 GALT ELECTRIC CONFIGURATION STUDIO

8.1 Overview

The interface card is discovered, configured and updated by the Galt Electric Configuration Studio PC application (refer to Figure 7). The studio requires a USB connection for discovery, configuration and firmware updates. To obtain the latest release of the Configuration Studio, refer to the [product web page](#) on the internet or contact technical support. The remainder of this section will provide only a brief introduction to the configuration concepts. For protocol specific configuration, refer to the relevant protocol section in section 10.

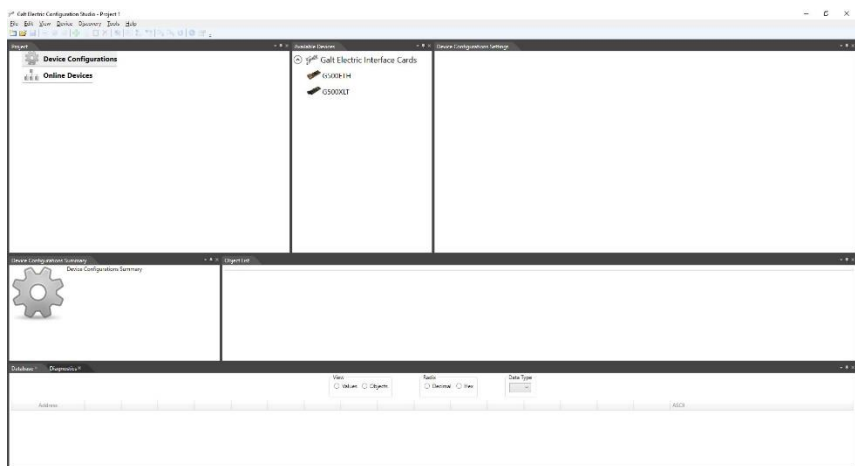


Figure 7: Galt Electric Configuration Studio

Creating a Device Configuration

A device can be added to the **Project** panel for configuration by first selecting the **Device Configurations** list heading and then:

- Double-clicking on the device in the **Available Devices** panel.
- Right-clicking on the device in the **Available Devices** panel and choosing **Add** from the context-sensitive menu.
- Hitting the <ENTER> key on the keyboard when the device is selected in the **Available Devices** panel.
- Dragging the device from the **Available Devices** panel into the **Project** panel.
- Selecting it and selecting **Add Selected Device** from the **Edit** menu.
- Selecting it and clicking the **Add** button in the toolbar.

The device will then be added to the list of **Device Configurations**.

Going Online with a Device

All connected devices are automatically added to the **Discovered Devices** panel. This panel is shown by selecting the **Online Devices** list heading in the **Project** panel. To go online with a device:

- Double-click on it in the **Discovered Devices** panel.
- Right-click on it in the **Discovered Devices** panel and choose **Go Online** from the context-sensitive menu.
- Hit the <ENTER> key on the keyboard when the device is selected in the **Discovered Devices** panel.
- Drag it from the **Discovered Devices** panel into the **Project** panel.
- Select it and select **Go Online with Device** from the **Edit** menu.

- Select it and click the **Go Online** button in the toolbar.

When the studio goes online with a device, its configuration is automatically read. While the studio is online with a device, it will appear in green text in the **Discovered Devices** panel. The studio may be online with multiple devices simultaneously. Note that the online configuration is read-only. The online configuration must first be uploaded into a project before it can be modified.

Uploading a Device's Configuration into a Project

The current configuration of an online device can be uploaded into the **Project** panel by selecting a device under the **Online Devices** list heading and then:

- Right-clicking on it and choosing **Upload Configuration** from the context-sensitive menu.
- Dragging it from the **Online Devices** heading into the **Device Configurations** heading.
- Selecting it and selecting **Upload Configuration to Project** from the **Device** menu.
- Selecting it and clicking the **Upload Configuration** button in the toolbar.

The device's configuration will then be added to the list of **Device Configurations**. Once the configuration is uploaded into the project, it may be modified.

Removing a Device Configuration from a Project

A configuration can be removed from a project by:

- Selecting the device in the **Project** panel and dragging it. A trash can icon will appear at the bottom of the **Project** panel, and dragging and dropping the device in the trash will remove it from the project.
- Hitting the <DELETE> key on the keyboard when the device is selected in the **Project** panel.
- Right-clicking on the device in the **Project** panel and choosing **Remove** from the context-sensitive menu.
- Selecting **Remove Selected Item** from the **Edit** menu when the device is selected.
- Clicking on the **Remove** button in the toolbar when the device is selected.

Going Offline with a Device

To go offline with a device:

- Select the device in the **Project** panel and drag it. A trash can icon will appear at the bottom of the **Project** panel, and dragging and dropping the device in the trash will go offline with it.
- Hit the <DELETE> key on the keyboard when the device is selected in the **Project** panel.
- Right-click on the device in the **Project** panel and choose **Go Offline** from the context-sensitive menu.
- Select **Go Offline with Device** from the **Edit** menu when the device is selected.
- Click on the **Go Offline** button in the toolbar when the device is selected.

Downloading a Configuration to a Device

To download a configuration to an online device, first select the device under the **Device Configurations** heading in the **Project** panel, and then navigate to **Device...Download Configuration to Device**. If the studio is currently online with only one compatible device, then the configuration will be downloaded to the online device. Otherwise, a device selection prompt is displayed to select which device to download the configuration to. Do not power off the device or interrupt the connection once the download is in progress as this may corrupt the firmware and/or the configuration.



Note Stop all other communication to the device when downloading.

Updating Firmware

The studio automatically manages firmware updates when going online with a device and downloading a configuration to a device. Download the latest studio from the [product web page](#) to obtain the latest firmware. Do not power off the device or interrupt the connection once the update is in progress as this may corrupt the firmware and/or the configuration.

Resetting an Online Device

To reset an online device, first select the device in the **Project** panel and then navigate to **Device...Reset Device**.

General Configuration Process

To configure a device, add the desired protocol(s) and configure any objects associated with the respective protocol(s). Any changes will take effect once the configuration is downloaded to a device.

Note that numeric values can be entered not only in decimal but also in hexadecimal by including "0x" before the hexadecimal number.

8.2 General Object Editing Activities

The following editing activities apply for all types of configuration objects and project elements.

Adding an Object

To add an object, click on an item (protocol driver or Node, for example) in the **Project** panel. Any available objects for that item will be listed in the **Available Objects** panel (the panel title depends on the currently-selected item). An object can then be added to the item by:

- Double-clicking on it.
- Right-clicking on it and choosing **Add** from the context-sensitive menu.
- Hitting the <ENTER> key on the keyboard when the object is selected.
- Dragging it into the **Project** panel.
- Selecting it and selecting **Add Selected Device** from the **Edit** menu.
- Selecting it and clicking the **Add** button in the toolbar.

The object's configurable fields can then be populated with valid values (where applicable).

Viewing an Object

In the **Project** panel, select a parent object to display a summary of all its child objects. For example, selecting a protocol driver will display the driver's configuration in the **Summary** panel and list of current objects in the **Object List** panel.

Updating an Object

To update an object, select the object in the **Project** panel and make any required changes in the **Settings** panel.

Deleting an Object

An object can be deleted by performing one of the three following actions:

- Selecting the object in the **Project** panel and dragging it. A trash can icon will appear at the bottom of the **Project** panel, and dragging the object to the trash will then delete it from the project.
- Hitting the <DELETE> key on the keyboard when the object is selected in the **Project** panel.
- Right-clicking on the object in the **Project** panel and choosing **Remove** from the context-sensitive menu.
- Selecting **Remove Selected Item** from the **Edit** menu when the object is selected.
- Clicking on the **Remove** button in the toolbar when the object is selected.

Note that this action cannot be undone. Deleting an object will also delete all of its child objects.

Copying and Pasting an Object

To copy an object, first click on an item in the **Project** panel. An object can then be copied by:

- Right-clicking on it and choosing **Copy** from the context-sensitive menu.
- Pressing the <CTRL+C> keys on the keyboard.
- Holding the <CTRL> key and dragging the item to the desired location in the **Project** panel.
- Dragging the item to a new location under a different parent object in the **Project** panel.
- Selecting **Copy Selected Item** from the **Edit** menu.

- Clicking on the **Copy** button in the toolbar.

To paste an object, first click on an item at the desired location in the **Project** panel. An object can then be pasted by:

- Right-clicking on it and choosing **Paste** from the context-sensitive menu.
- Pressing the <CTRL+V> keys on the keyboard.
- Dropping an item onto the desired location in the **Project** panel after holding the <CTRL> key and dragging the item.
- Dropping an item onto a new location under a different parent object in the **Project** panel after dragging the item.
- Selecting **Paste Item** from the **Edit** menu.
- Clicking on the **Paste** button in the toolbar.

After pasting an object, the object's configurable fields can then be modified with valid values (where applicable).

Note that the studio allows you to copy and paste items between different locations, including different devices. This is useful for copying partial configurations from one device to another.

Reordering Objects

Objects can be reordered in the **Project** panel by dragging the item to the desired location. If the item is dragged outside of the items in the project tree, it will be moved to the end.

8.3 Device Settings

The following fields can be configured for a device. To view or edit device settings, click on the device in the **Project** panel. The settings are then available in the **Settings** panel.

Device Description

Each device added to a project can be individually tagged with a unique description string of up to 32 characters in length. This allows the devices within a project or an automation system to be clearly identifiable with their location or functional purpose.

Default Network Protocol

Select the network protocol which will be activated by default after the configuration is downloaded to the device. If only one network protocol is configured, or the default protocol is irrelevant, set this to **Automatic**.

Inverter Timeout

Defines the maximum number of milliseconds without receiving valid communications from the inverter before triggering a Host Communication Error. To disable inverter timeout detection, set this field to 0.

8.3.1 Status LED Settings

The device's status LED is software-controlled and can be configured to indicate a wide range of information to the user, providing useful insight into the operation of the device. These settings define the behavior of the device's status LED.

Status LED Control

Selects the method for controlling the device's status LED. Regardless of the option selected, upon startup, the status LED will flash the green, red, green, red startup sequence and then flash green rapidly while the card is synchronizing parameters with the inverter. Additionally, if an internal error occurs, the status LED will always flash red indicating the error code.

Default

This option is the default behavior of the status LED. The LED is solid green when power is applied to the device. The LED flashes green when a USB connection has been established between the device and a PC.

Port Activity

This option allows the selected port's TX and RX activity to be indicated by the status LED. For each LED cycle, if the port has transmitted any bytes since the last cycle, the status LED will light green for half of the cycle. If the port has received any bytes since the last cycle, the status LED will light red for half of the cycle.

Register Value

This option configures the status LED to be fully controlled by a value located in the device's internal register database. This enables the status LED to be directly controlled via communications or by internal logic applied to data stored in the device's database. Table 33 lists the supported LED states.

Port

Selects the port for which the TX and RX activity will be indicated by the status LED.

Note that this option only applies when the Port Activity option is selected for the Status LED Control setting.

Function Code

Defines the inverter function code used to control the status LED.

Note that this option only applies when the Register Value option is selected for the Status LED Control setting.

8.4 Process Data Configuration

The card supports user-configurable process data. Process data is exchanged between the card and inverter at high speed, every communication cycle. Command process data is sent to the inverter from the card and status process data is received by the card from the inverter. There are a maximum of 11 command process data parameters (Command Process Data 0 - 10) and 11 status process data parameters (Status Process Data 0 - 10) that can be exchanged between the card and the inverter (refer to sections 5.3.3 and 5.3.7 for details on these parameters). The selection of which data is mapped to these process data parameters is performed by adding a process data assignment object.



Inverter parameters P15.02 - P15.23 configure which process data is exchanged between the card and inverter. The card automatically overwrites these parameters on startup for all configured process data assignments.

8.4.1 Process Data Assignment Object

A process data assignment object defines the data assigned to a specific process data parameter.

Description

This field is strictly for user reference: it is not used at any time by the device.

Command/Status Parameter

Selects the command/status process data parameter that will contain the value for the assigned data.

Assignment

Selects the data assignment for the configured process data parameter.

8.5 USB Virtual COM Port Settings

The card can be configured to enumerate as a USB virtual COM port, providing direct serial communications between the card and a PC through the USB connection. The COM port can be used for various tasks, depending on the selected mode. This section details the different functions of the virtual COM port.

Mode

Select the desired mode for how the USB virtual COM port will be used. The available options are detailed below.

Serial Pass-Through

Select this option to cause the card to behave as a USB to serial converter. Any data sent to the USB virtual COM port will be sent on the physical serial port and any data received by the physical serial port will be received from the USB virtual COM port. Note that while the card is in this mode all other functionality of the card is disabled, regardless of other configuration settings.

Serial Redirect

Select this option to redirect communications from the selected serial port to the USB virtual COM port. By selecting this option, the card will communicate with the PC over the virtual COM port using the settings configured on the associated serial port. This allows the card to communicate with the PC using any of the supported serial port protocols. Note that the physical serial port is disabled when the card is configured in this mode.

Serial Sniffer

Select this option to sniff the received and transmitted packets on the selected serial port and output the data to the virtual COM port. When this mode is selected, the card will attempt to output every packet that the protocol driver configured on the serial port receives and transmits.

Because the sniffer operates independently from the physical serial port (so as not to impact communications), there may be times when the sniffer cannot output a received or transmitted packet due to the USB connection being unable to output characters faster than they are exchanged on the physical serial port. When this occurs, the sniffer will output the characters "ERR: Sniffer Packet Overflow" or "ERR: Sniffer Buffer Overflow". Additionally, the sniffer is able to detect receive errors on the serial port such as parity, overrun, and framing errors. If a receive error occurs on one or more characters of a packet, the sniffer will output the characters "ERR: Receive Error".

Note that because the serial sniffer mode captures packets at the protocol driver level, a protocol must be configured on the selected serial port to output data to the USB virtual COM port. For convenience, there is a special "USB Serial Sniffer Settings" protocol selection to configure the serial port for sniffing only.

Serial Port

Select the desired serial port to target for use with the USB virtual COM port.

Sniffer Output Format

Select the desired output format of the serial sniffer data. The formatted data option outputs the captured data as ASCII text characters and includes annotations for whether the packet was received or transmitted, as well as a relative timestamp of when the packet was received or transmitted. The raw data option outputs the captured data as unmodified, binary characters.

8.6 USB Serial Capture Window

The USB Serial Capture Window allows you to connect to a card's USB Virtual COM port to view and save network packets captured by the card. The card's USB Virtual COM port must be configured for Serial Sniffer mode and the Sniffer Output Format must be set to Formatted Data.

When connected, the capture window will display the card's most recent received and transmitted packets. All packets captured during the duration of the session may be saved once the session has ended, even though they all may not be displayed in the window. The status bar at the bottom of the window tracks the duration of the connection as well as the total number of packets the card has received and transmitted.

To open the USB Serial Capture Window, select **USB Serial Capture Window...** from the **Tools** menu.

Capturing Packets

To begin capturing packets, the card must first be configured with the appropriate USB Virtual COM port settings as described above. Once configured, the card will appear in the **COM Port** selection box.

Select the desired device from this drop down and connect to the device. To connect to the device, perform one of the following actions:

- Select **Connect** from the **Connection** menu.
- Click on the **Connect** button in the toolbar.

Note that connecting to a device will clear the capture log automatically.

Clearing the Capture Log

All captured data may be cleared at any time while connected to a device or after disconnecting from a device. This will also reset the connection time duration and all counters. To reset all captured data, perform one of the following actions:

- Select **Clear Log** from the **Edit** menu.
- Click on the **Clear Log** button in the toolbar.
- Hit the <DELETE> key on the keyboard.
- Right click on the capture output and select **Clear Log**.

Pausing the Display

While capturing, the output window will display only the most recent packets. Therefore, as new packets are captured and displayed in the window, old packets are removed from the display. At any time during capturing, the display updating may be paused so that no packets are added or removed. To pause the display, perform one of the following actions:

- Select **Pause Display** from the **Display** menu.
- Click on the **Pause Display** button in the toolbar.
- Right click on the capture output and select **Pause Display**.

Note that even though the display does not update when paused, packets are still being captured in the background.

Ending a Capture Session

The capture session is ended by disconnecting from the selected device. To disconnect from the device, perform one of the following actions:

- Select **Disconnect** from the **Connection** menu.
- Click on the **Disconnect** button in the toolbar.

Saving the Captured Data

Once a capture session has ended, the entire captured data may be saved. The data can be saved either as a Wireshark capture file or as a plain text document.

Wireshark Capture File

The captured data can be saved as a file which can be opened, decoded, and analyzed by Wireshark. Wireshark is a free network protocol analyzer and is available at <http://www.wireshark.org/>.

Any protocol capture may be viewed with Wireshark. However, Wireshark currently only supports decoding BACnet MS/TP packets and has limited support for Modbus RTU.

To save the captured data as a Wireshark capture file, perform one of the following actions:

- Select **Save As Wireshark Capture...** from the **File** menu.
- Click on the **Save As Wireshark Capture...** button in the toolbar.
- Hit the <CTRL+S> keys on the keyboard.

Text Document

The captured data can also be saved as a plain text document. To save the captured data as a text document, perform one of the following actions:

- Select **Save As Text...** from the **File** menu.
- Click on the **Save As Text...** button in the toolbar.
- Hit the <CTRL+SHIFT+S> keys on the keyboard.

8.7 Port Diagnostics

The Port Diagnostics window provides real-time monitoring of the number of bytes sent and received by the selected port. In addition, the Port Diagnostics also displays receive errors detected by the port.

8.8 Batch Update Mode

The Configuration Studio supports a batch update mode for quickly updating firmware, and optionally, the configuration on all discovered devices without user interaction. While in batch update mode, the studio will automatically go online with a card, update the firmware, update the configuration if a matching configuration is found in the project, and then go offline with the card. It will do this for all discovered devices while in this mode. For each discovered device, the studio creates a log entry in the batch update log detailing the actions performed on the card.

Entering Batch Update Mode from within the Studio

To start batch update mode when the studio is open, select **Start Batch Update Mode** from the **Tools** menu. After the studio has entered batch update mode, pressing the ESC key will exit batch update mode. If any devices were discovered while in batch update mode, the studio will display a prompt to view the batch update log.

Launching the Studio in Batch Update Mode

The batch update mode can also be started when the studio is launched by using the “-b” or “-B” command line switch, and optionally, specifying a project file path to load. For example, the command line options “-b MyProject.gcsproj” will load the project titled “MyProject” and start batch update mode. When batch update mode is entered using this method, the user cannot exit batch update mode using the ESC key.

Note that the command line options can also be used with a custom shortcut by appending them to the executable path in the **Target** field of the shortcut. This would allow a user to double click on the shortcut to launch the studio in batch update mode.

Viewing the Batch Update Log

After the studio has updated a card while in batch update mode, a log is available that can be accessed by selecting **Open Batch Update Log** from the **Help** menu. The log details the actions that the studio performed on discovered devices during the last batch update session.

At the end of the log, the studio records statistics for the batch update session. The statistics include the following information:

Devices Discovered

The total number of devices discovered while in batch update mode.

Successful

The total number of devices that were updated successfully.

Failed

The total number of devices that the studio failed to update.

Not Updated

The total number of devices that were not updated. This can occur if a device is already up to date, or if a device has limited network connectivity and cannot be updated.

Firmware Updated

The total number of firmware updates performed.

Configuration Updated

The total number of configuration updates performed.

Errors

The total number of devices that encountered an error while being updated. Note that this does not necessarily imply that the device failed to update.

8.9 Internal Logic Settings

8.9.1 Initial Persistent Values

8.9.1.1 Overview

The card can be configured to write initial values to persistent user parameters using initializer objects. Persistent memory is initialized only once after a configuration has been downloaded to the device. This mechanism is useful for providing initial factory values for parameters mapped to the device's persistent memory. For more information on the card's persistent user parameters, refer to section 5.3.14.

8.9.1.2 Initializer Object Configuration

An initializer object is used to provide an initial value for parameters mapped to the persistent memory locations in the device's database. When persistent memory is initialized, the initializer objects are parsed and the designated value is written to the corresponding persistent parameters. To add an initializer object to a device, select the device in the **Project** panel, then add **Internal Logic...Initial Persistent Values...Initializer Object**. The following paragraphs describe the configurable fields of an initializer object:

Description

This field is strictly for user reference: it is not used at any time by the device.

Function Code

Enter the function code (refer to section 5) that this object will write to.

Value

Enter the value that the function code encompassed by this initializer object will be written to when the persistent memory is initialized.

8.9.2 Fail-safe Values

8.9.2.1 Overview

The card can be configured to perform a specific set of actions when network communications are lost (timeout event). This allows each inverter parameter to have its own unique "fail-safe" condition in the event of network interruption. Support for this feature varies depending on the protocol: refer to the protocol-specific section of this manual for further information.

There are two separate elements that comprise the timeout configuration:

- The timeout time
- Timeout Object configuration

8.9.2.2 Timeout Time

The timeout time is the maximum number of milliseconds for a break in network communications before a timeout will be triggered. This timeout setting is configured at the protocol level as part of a driver's configuration, and used by the protocol drivers themselves to determine abnormal loss-of-communications conditions. These conditions then trigger timeout processing events. If it is not desired to have a certain protocol trigger timeout processing events, then the protocol's timeout time may be set to 0 (the default value) to disable this feature.

8.9.2.3 Timeout Object Configuration

A timeout object is used as part of the timeout processing to set certain parameters to "fail-safe" values. When a timeout event is triggered by a protocol, the timeout objects are parsed and written to the corresponding function code(s). The timeout object(s) will be executed sequentially from first to last. To add a timeout object, select the device in the **Project** panel, then add **Internal Logic...Fail-safe Values...Timeout Object**. The following paragraphs describe the configurable fields of a timeout object:

Description

This field is strictly for user reference: it is not used at any time by the device.

Function Code

Enter the function code (refer to section 5) that this object will write to.

Value

Enter the “fail-safe” timeout value that the function code encompassed by this timeout object will be automatically written with upon processing a timeout event triggered by a protocol.

8.9.2.4 Fail-safe Example

This example will demonstrate how to add one timeout object which will assign a value of 0 to function code P100.01 (command word). In the **Project** panel, select the device and add **Internal Logic...Fail-safe Values...Timeout Object** as shown in Figure 8. The red error indicators are normal at this stage as the **Timeout Object Settings** have not yet been configured.



Figure 8: Timeout Object Project Panel

Next, configure the **Timeout Object Settings** as shown in Figure 9.

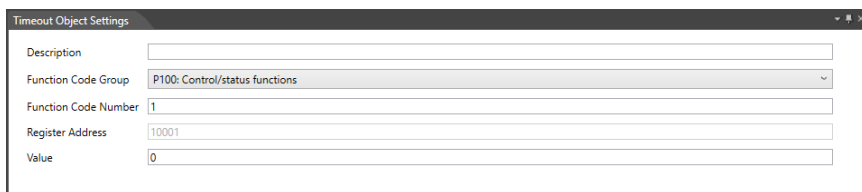


Figure 9: Timeout Object Settings

The example is complete.

8.9.3 Database Logic

8.9.3.1 Overview

A variety of database logic operations are included which provide PLC-style manipulation of database values. Categories such as logical, arithmetic and filtering operations allow for autonomous control over value modification and data movement within the database. High-level signal conditioning is also realizable via the construction of compound formulas derived from the elemental building block operations provided. To add database logic operations to a device, select the device in the **Project** panel, then add **Internal Logic...Database Logic**.

Database logic operations are executed in sequential order, according to the ordinal position in which the operations are listed in the **Project** panel under the **Database Logic** heading.

Some notes of interest for the database logic operations are as follows:

All Database Logic Operations

- All inputs to an operation may either be a value located in the internal database or a constant value.
- A floating-point “Multiplier” field is available on each database-sourced input and on the output which allows the inputs to be scaled prior to operation execution, and the result to be scaled after

operation execution. The input is multiplied by the input multiplier, and the result is divided by the output multiplier.

- All operations can be dynamically enabled/disabled using an optional "Enable Trigger" element (refer to section 8.9.3.3 for more information on Enable Trigger behavior.)
- The outputs of all operations must be stored in the internal database.

Logical Operations

- The *Not*, *And*, *Or*, and *Exclusive Or* operations can be performed on either a bitwise or logical basis, depending on the selection of the "Operation Type". When a logical operation type is chosen, non-zero input values are considered to be "true" and zero input values are considered to be "false". The output value of the logical operation will then be written to the database as "1" for true and "0" for false.
- The *Copy* operation outputs the input value.
- The *Bit Copy* operation outputs the value of a single bit from the input database location to a single bit in the output database location. No other bits in the output database location are modified by this operation.
- The *Indirect Copy* operation outputs the value at the database location specified by the input source to the database location specified by the output destination. This operation can be used to access different database locations dynamically. It could also be used to create reusable database logic subroutines by selecting a different input and output location for the subroutine during each execution cycle.
- The *Shift* operation outputs the input value bit-shifted by the shift amount.
- The *Compare* operation outputs a "1" if the comparison evaluates to true, otherwise it outputs a "0".
- The *Flag Test & Set* operation tests if the bit flags specified in the input mask are set in the input value and sets the bit flags specified in the output mask in the output value. This operation can test for ALL flags set/cleared or ANY flags set/cleared. If the flag test evaluates as true, all bit flags specified in the output mask in the output value are set, otherwise the flags are cleared. Only the bits specified in the output mask in the output value are modified by this operation.
- The *Value Change Detection* operation outputs a "1" if a change is detected in the input value between the last execution cycle and the current execution cycle, otherwise it outputs a "0".
- The *Multiplexer* operation outputs one of its two inputs, depending on the selection. If *Selection* is zero, *Input 1* is output. If *Selection* is non-zero, *Input 2* is output.
- The *Byte Reverse* operation reverses the byte order of the input value and outputs the result.

Arithmetic Operations

- The *Add* operation calculates the expression $[Input\ 1] + [Input\ 2]$.
- The *Subtract* operation calculates the expression $[Input\ 1] - [Input\ 2]$.
- The *Multiply* operation calculates the expression $[Input\ 1] \times [Input\ 2]$.
- The *Divide* operation calculates the expression $[Input\ 1] / [Input\ 2]$.
- The *Modulo* operation calculates the expression $[Input\ 1] \bmod [Input\ 2]$.
- The *Exponential* operation calculates the expression $[Input\ 1]^{Exponent}$. "Input 1" can be a database value, a constant value, or e (exponential function).
- The *Nth Root* operation calculates the expression $\sqrt[Degree]{Input\ 1}$.
- The *Logarithm* operation calculates the expression $\log_{Base}(Input\ 1)$. "Base" can be a database value, a constant value or e (natural logarithm).
- The *Random* operation outputs a random number between *Input 1* and *Input 2*. Note that the operation is limited to producing only 32,768 unique values.
- The *Divide*, *Exponential*, *Nth Root* and *Logarithm* operations output an integer-rounded value when an integer data type is used.

Trigonometric Operations

- The *Sine* operation calculates the expression $\sin(Input\ 1)$, where *Input1* is in radians.
- The *Cosine* operation calculates the expression $\cos(Input\ 1)$, where *Input1* is in radians.
- The *Tangent* operation calculates the expression $\tan(Input\ 1)$, where *Input1* is in radians.

- The *Arc Sine* operation calculates the expression $\sin^{-1}(\text{Input } 1)$, where the output is in radians.
- The *Arc Cosine* operation calculates the expression $\cos^{-1}(\text{Input } 1)$, where the output is in radians.
- The *Arc Tangent* operation calculates the expression $\tan^{-1}(\text{Input } 1)$, where the output is in radians.

Filtering Operations

- The *Debounce Filter* and *Hysteresis Filter* operations are functionally identical with the single exception that the *Debounce Filter* does not use a "Value Tolerance" (it is fixed at 0).
- In order for the output of the *Debounce Filter* or *Hysteresis Filter* to change (i.e. reflect the input value), "Input 1" must first change to a value outside of the "Value Tolerance" range and then remain within the "Value Tolerance" range of the new value for the entire "Stable Time".

8.9.3.2 Database Logic Settings

Scan Rate

Defines the scan cycle time in milliseconds (50ms minimum) of the database logic processing task. All operations are evaluated for execution in sequential order at this frequency. Note that this does not necessarily mean that each operation is guaranteed to execute every scan cycle: only that it will be evaluated as to whether or not it should execute. Namely, if an "Enable Trigger" element is added to an operation, then the trigger must evaluate to "true" for the operation to execute during that scan cycle. Refer to section 8.9.3.3 for more information on Enable Trigger behavior.

8.9.3.3 Enable Trigger

Each database logic operation can optionally include an "Enable Trigger" element, which provides dynamic conditional execution capabilities. By default (i.e. if an enable trigger element is not added to the operation), each operation is automatically triggered to execute every scan cycle. If it is desired for an operation to execute conditionally, however, then an enable trigger element can be added to it. The enable trigger element defines an "Enable Value", which specifies a byte-size trigger value that can reside at any location in the internal database. When implemented, the enable value is evaluated every scan cycle: if this value is non-zero (or zero when the "Inverted" Trigger Option is used), the operation will execute.

The enable value itself can be modified by any communication driver currently running on the device, which enables networked devices to dynamically control the execution of database logic operations. The enable value can also be the output result of other database logic operations. While the output of any database operation can be used for this purpose, such a scenario may most typically use the output of a "compare" operation in order to control whether or not other operations should execute (e.g. execute a certain operation only when some process variable is greater than a certain value, etc.) Allowing the conditional execution of database logic operations to be based on data values obtained via communications or as a result of other database logic operations enables the construction of flexible, hierarchical and dynamic data evaluation and manipulation engines.

Enable Function Code

Specifies the function code that will be used to enable the associated database logic operation.

8.9.3.3.1 Trigger Options

The enable trigger can perform basic logic on the enable value to determine if an operation should execute using a variety of trigger options. These settings determine what logic should be applied to the enable value when evaluating whether or not the operation should execute.

Inverted

Specifies whether the enable logic should be inverted. This applies to both the evaluation of whether or not the operation should execute as well as resetting the enable value when the auto reset option is used.

Auto Reset

Allows the enable value to be automatically reset upon completion of the operation. The actual value written to the enable value depends on the other trigger options selected. If no options are selected, a value of 0 is written to the enable value. If the inverted option is used, a value of 1 is written to the enable value. If the bitmask option is used, each bit selected in the bitmask is written to a 0 (or a 1 if the inverted option is used) in the enable value.

Bitmask

If this option is used, it selects which bits in the enable value to evaluate. Every selected bit in the enable value must be 1 (or 0 when the inverted option is used) for the operation to execute.

8.10 Manage Device Parameters

The function code and register address of all available parameters can be viewed using the **Manage Device Parameters** window. This window is found by:

- Right-clicking on the device in the **Project** panel and choosing **Manage Parameters...** from the context-sensitive menu.
- Selecting the device in the **Project** panel and navigating to **Device...Manage Device Parameters...**



Note The parameter list serves as a useful reference when configuring objects.

Function Code	Register Address	Description
P00: Basic functions 19 Total		
P00.00	0	Speed control mode
P00.01	1	Running command channel
P00.02	2	Communication running command channel
P00.03	3	Max. output frequency
P00.04	4	Upper limit of running frequency
P00.05	5	Lower limit of running frequency
P00.06	6	A frequency command selection
P00.07	7	B frequency command selection
P00.08	8	Reference object of B frequency command
P00.09	9	Combination mode of setting source
P00.10	10	Set frequency via keypad
P00.11	11	Acceleration time 1
P00.12	12	Deceleration
P00.13	13	Running direction
P00.14	14	Carrier frequency setup
P00.15	15	Motor parameter autotuning
P00.16	16	AVR function
P00.17	17	Reserved
P00.18	18	Function parameter restoration
P01: Start/stop control 35 Total		
P01.00	100	Running mode of start
Total: 1320		

Figure 10: Manage Device Parameters

8.11 Monitor Device Parameters

The inverter's parameter values can be monitored and commanded in real-time (refer to Figure 11). The **Monitor Device Parameters** window is found by:

- Right-clicking on the online device in the **Project** panel and choosing **Monitor Parameters...** from the context-sensitive menu.
- Selecting the online device in the **Project** panel and navigating to **Device... Monitor Device Parameters...**

Radix and data type selections are available at the top of the window to select the display format of the **Value** column. The **Bits 15...0** column shows the current value of each bit in the parameter's value, starting at bit 15 on the left and continuing to bit 0 on the right. The bits are grouped into 4-bit nibbles for readability purposes.

A filter option, found at the top of the window, can be used to filter which parameters are shown. The filter function compares the filter text to each parameter's description, group name, parameter number, and communications number to display matching parameters. To use the filter function, simply type a word, or portion of a word, into the filter entry box. To reset the filter, click the **X** button to the right of the filter entry box. The filtering function is case insensitive.

Parameter Number	Communications Number	Description	Value	Bits 15...0
P00: Basic functions 19 Total				
P00.00	0	Speed control mode	2	0000 0000 0000 0010
P00.01	1	Running command channel	2	0000 0000 0000 0010
P00.02	2	Communication running command channel	1	0000 0000 0000 0001
P00.03	3	Max. output frequency	6000	0001 0111 0111 0000
P00.04	4	Upper limit of running frequency	6000	0001 0111 0111 0000
P00.05	5	Lower limit of running frequency	0	0000 0000 0000 0000
P00.06	6	A frequency command selection	7	0000 0000 0000 0111
P00.07	7	B frequency command selection	15	0000 0000 0000 1111
P00.08	8	Reference object of B frequency command	0	0000 0000 0000 0000
P00.09	9	Combination mode of setting source	0	0000 0000 0000 0000
P00.10	10	Set frequency via keypad	6000	0001 0111 0111 0000
P00.11	11	Acceleration time 1	100	0000 0000 0110 0100
P00.12	12	Deceleration	100	0000 0000 0110 0100
P00.13	13	Running direction	0	0000 0000 0000 0000
P00.14	14	Carrier frequency setup	80	0000 0000 0101 0000
P00.15	15	Motor parameter autotuning	0	0000 0000 0000 0000
P00.16	16	AVR function	1	0000 0000 0000 0001
P00.17	17	Reserved	0	0000 0000 0000 0000

Total: 1320 Connected

Figure 11: Monitor Device Parameters

8.12 Backup and Restore Parameters

The parameter values can be backed up from the inverter and restored to the inverter (refer to Figure 12 and Figure 13). This allows for easy inverter cloning. The backup parameter values are stored as a CSV file. A parameter value can be excluded from the list by disabling the corresponding checkbox. The parameter value can also be modified before the backup and restore is executed. Note that backup and restore does not modify the parameter list (refer to section 8.9.3). The backup and restore parameter configurations are found by:

- Right-clicking on the device in the **Project** panel and choosing **Backup Parameters...** or **Restore Parameters...** from the context-sensitive menu.
- Selecting the device in the **Project** panel and navigating to **Device...Backup Parameters from Device...** or **Restore Parameters to Device...**

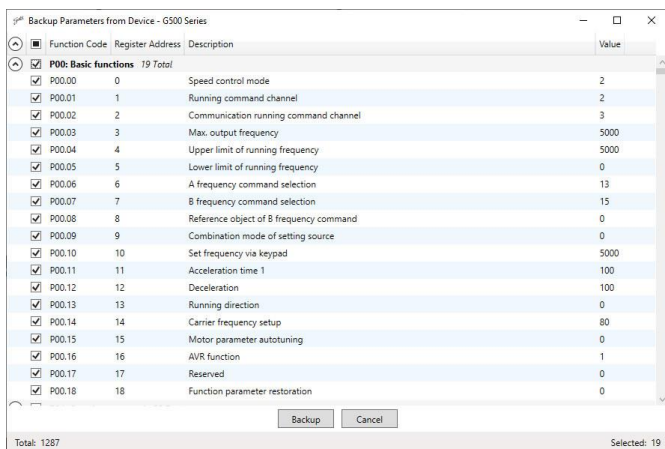


Figure 12: Backup Parameters

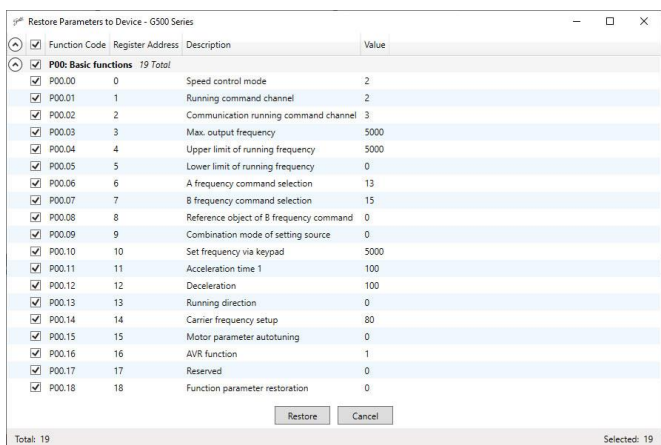


Figure 13: Restore Parameters

8.13 Restore Factory Settings

The interface card (connected via USB) can be restored to the factory settings. Note that the filesystem will be reformatted, which will destroy all custom modifications and configurations. Please backup the configuration before executing this feature. The factory settings can be restored by:

- Right-clicking on the device in the **Project** panel and choosing **Restore Factory Settings**.
- Selecting the device in the **Project** panel and navigating to **Restore Factory Settings**.

8.14 Database

To interact in real-time with inverter parameters, select the online device in the **Project** panel and then select the **Database** panel. Parameter values can be edited by double-clicking the desired location in the database. If the **Database** panel is not visible, it can be enabled via **View...Database**.

8.15 Help

Links to videos and documents can be found in the **Help** menu. Please review these links before contacting technical support for more in-depth assistance.

9 FIRMWARE

9.1 Overview

The interface card's embedded firmware can be updated in the field. Firmware updates may be released for a variety of reasons, such as custom firmware implementations, firmware improvements and added functionality as a result of user requests. Additionally, it may be necessary to load different firmware onto the unit in order to support various protocols. In order to ensure that the firmware update is successful, and in the interest of equipment and personnel safety, it is strongly recommended to stop all of the card's production activities prior to initiating the firmware update procedure.



Note Failure to follow the firmware update procedure could result in corrupt firmware!

9.2 Update Procedure

1. Always back up your configuration to a PC for later recovery if necessary.
2. Download and install the latest Network Parameter Utility or Configuration Studio, which can be obtained from the [product web page](#).
3. Please be sure to read the firmware release notes and updated user's manual for any important notices, behavior precautions or configuration requirements prior to updating your firmware.
4. Ensure that the device is in a safe state prior to initiating the firmware update. The card may be temporarily inaccessible during the firmware update process.
5. Connect a USB cable between the card and the PC and open the NPU or Configuration Studio. If the software contains newer firmware, it will automatically prompt you to update the firmware. Proceed with the firmware update.
6. Once the firmware update process has started, do not interrupt the card as this may corrupt the firmware. Do NOT manually power-cycle the inverter or reboot the card. Do NOT disturb the USB connection.
7. After the firmware update has been completed, the card will reset automatically. When the card boots up again, it will be running the new application firmware, which can be confirmed by observing the version displayed in the NPU or Configuration Studio.

10 PROTOCOL-SPECIFIC INFORMATION

This section will discuss topics that are specific to each of the supported protocols.

10.1 BACnet MS/TP

- The interface card supports the BACnet MS/TP protocol over RS-485 at baud rates of 9600, 19200, 38400, 57600, 76800, and 115200.
- Supports up to 100 simultaneous COV subscriptions.

10.1.1 Protocol Implementation Conformance Statement (PICS)

BACnet Protocol

Date:	March 30, 2020
Vendor Name:	ICC, Inc.
Product Name:	Galt Electric G500
Product Model Number:	G500XLT
Application Software Version:	V1.000
Firmware Revision:	V1.000
BACnet Protocol Revision:	12
Product Description:	

The G500XLT RS-485 multiprotocol communication interface allows information to be transferred seamlessly between a G500 series inverter and several different RS-485-based fieldbus networks.

BACnet Standard Device Profile (Annex L):

- ☐ BACnet Operator Workstation (B-OWS)
- ☐ BACnet Building Controller (B-BC)
- ☐ BACnet Advanced Application Controller (B-AAC)
- ☒ BACnet Application Specific Controller (B-ASC)
- ☐ BACnet Smart Sensor (B-SS)
- ☐ BACnet Smart Actuator (B-SA)

BACnet Interoperability Building Blocks Supported (Annex K):

- ☒ Data Sharing – ReadProperty-B (DS-RP-B)
- ☒ Data Sharing – ReadPropertyMultiple-B (DS-RPM-B)
- ☒ Data Sharing – WriteProperty-B (DS-WP-B)
- ☒ Data Sharing – WritePropertyMultiple-B (DS-WPM-B)
- ☒ Data Sharing – COV-B (DS-COV-B)
- ☒ Device Management – Dynamic Device Binding-B (DM-DDB-B)
- ☒ Device Management – Dynamic Object Binding-B (DM-DOB-B)
- ☒ Device Management – DeviceCommunicationControl-B (DM-DCC-B)
- ☒ Device Management – ReinitializeDevice-B (DM-RD-B)

Segmentation Capability:

None

- | | |
|---|-------------------|
| <input type="checkbox"/> Able to transmit segmented message | Window Size _____ |
| <input type="checkbox"/> Able to receive segmented message | Window Size _____ |

Standard Object Types Supported:

See "Object Types/Property Support Table".

Data Link Layer Options:

- ☐ BACnet IP, (Annex J)
- ☐ BACnet IP, (Annex J), Foreign Device
- ☐ ISO 8802-3, RS-485 (Clause 7)
- ☐ ANSI/ATA 878.1, 2.5 Mb. ARCNET (Clause 8)

- ☐ ANSI/ATA 878.1, RS-485 ARCNET (Clause 8), baud rate(s) _____
- ☒ MS/TP master (Clause 9), baud rate(s): 9600, 19200, 38400, 57600, 76800, 115200
- ☐ MS/TP slave (Clause 9), baud rate(s): _____
- ☐ Point-To-Point, EIA 232 (Clause 10), baud rate(s): _____
- ☐ Point-To-Point, modem, (Clause 10), baud rate(s): _____
- ☐ LonTalk, (Clause 11), medium: _____
- ☐ Other: _____

Device Address Binding:

Is static device binding supported? (This is currently for two-way communication with MS/TP slaves and certain other devices.) ☐ Yes ☒ No

Networking Options:

- ☐ Router, Clause 6 - List all routing configurations
- ☐ Annex H, BACnet Tunneling Router over IP
- ☐ BACnet/IP Broadcast Management Device (BBMD)
- Does the BBMD support registrations by Foreign Devices? ☐ Yes ☐ No

Networking Security Options:

- ☒ Non-secure Device – is capable of operating without BACnet Network Security
- ☐ Secure Device – is capable of using BACnet Network Security (NS-SD BIBB)
- ☐ Multiple Application-Specific Keys:
- ☐ Supports encryption (NS-ED BIBB)
- ☐ Key Server (NS-KS BIBB)

Character Sets Supported:

Indicating support for multiple character sets does not imply that they can all be supported simultaneously.

- | | | |
|---|---|-------------------------------------|
| <input checked="" type="checkbox"/> ISO 10646 (UTF-8) | <input type="checkbox"/> IBM™/Microsoft™ DBCS | <input type="checkbox"/> ISO 8859-1 |
| <input type="checkbox"/> ISO 10646 (UCS-2) | <input type="checkbox"/> ISO 10646 (UCS-4) | <input type="checkbox"/> JIS X 0208 |

If this product is a communication gateway, describe the types of non-BACnet equipment/networks(s) that the gateway supports: N/A

Table 24: BACnet Device Object Types/Properties Supported

Property	Object Type
	Device
Object Identifier	W
Object Name	W
Object Type	R
System Status	R
Vendor Name	R
Vendor Identifier	R
Model Name	R
Firmware Revision	R
Application Software Version	R
Protocol Version	R
Protocol Revision	R
Protocol Services Supported	R
Protocol Object Types Supported	R
Object List	R
Max APDU Length Accepted	R
Segmentation Supported	R
APDU Timeout	W (10...65535)
Number Of APDU Retries	W (0...10)
Max Master	W (1...127)
Max Info Frames	W (1...100)
Device Address Binding	R
Database Revision	R
Active COV Subscriptions	R

R – readable using BACnet services

W – readable and writable using BACnet services

Table 25: BACnet Binary Object Types/Properties Supported

Property	Object Type		
	Binary Input	Binary Output	Binary Value
Object Identifier	R	R	R
Object Name	R	R	R
Object Type	R	R	R
Present Value	R	W	W
Status Flags	R	R	R
Event State	R	R	R
Reliability	R	R	R
Out Of Service	R	R	R
Priority Array		R	R
Relinquish Default		R	R
Polarity	W	W	
Active Text	R	R	R
Inactive Text	R	R	R

R – readable using BACnet services

W – readable and writable using BACnet services

Table 26: BACnet Analog Object Types/Properties Supported

Property	Object Type		
	Analog Input	Analog Output	Analog Value
Object Identifier	R	R	R
Object Name	R	R	R
Object Type	R	R	R
Present Value	R	W	W
Status Flags	R	R	R
Event State	R	R	R
Reliability	R	R	R
Out Of Service	R	R	R
Units	R	R	R
Priority Array		R	R
Relinquish Default		R	R
COV Increment	W	W	W

R – readable using BACnet services

W – readable and writable using BACnet services

Table 27: BACnet Multi-state Object Types /Properties Supported

Property	Object Type		
	Multi-state Input	Multi-state Output	Multi-state Value
Object Identifier	R	R	R
Object Name	R	R	R
Object Type	R	R	R
Present Value	R	W	W
Status Flags	R	R	R
Event State	R	R	R
Reliability	R	R	R
Out Of Service	R	R	R
Number Of States	R	R	R
Priority Array		R	R
Relinquish Default		R	R

R – readable using BACnet services

W – readable and writable using BACnet services

10.1.2 Default Supported Objects

This section will describe the default objects. Since the objects are configurable, the system integrator is responsible for managing, maintaining, and documenting the actual configuration.



Note Always use the studio to confirm the configuration before commissioning the device.

Table 28: Analog Input Object Instance Summary

Instance ID	Object Name	Description	Function Code	Units
AI1	Run Freq	Running frequency	P100.70	Hertz
AI2	Freq Ref	Frequency reference	P100.71	Hertz
AI3	Bus Voltage	DC bus voltage	P100.72	Volts
AI4	Output Voltage	Output voltage	P100.73	Volts
AI5	Output Current	Output current	P100.74	Amps
AI6	Output Torque	Actual output torque	P100.75	Percent
AI7	Output Power	Actual output power	P100.76	Percent
AI8	Fault Code	Fault code	P100.78	No Units
AI9	PID Ref	PID reference	P100.79	Percent
AI10	PID Fbk	PID feedback	P100.80	Percent
AI11	Torque Ref	Torque reference	P17.15	Percent
AI12	Operation Mode	Operation mode	P100.53	No Units
AI13	Motor Group Fbk	Motor group feedback	P100.52	No Units
AI14	Operation Time	Current running time	P17.26	Minutes
AI15	Mtr Power Factor	Motor power factor	P17.25	No Units
AI16	Power Cons Hi	High word of inverter power consumption	P07.15	Megawatt-hours
AI17	Power Cons Lo	Low word of inverter power consumption	P07.16	Kilowatt-hours
AI18	Fault Hist 1	Fault History 1 (most recent)	P07.27	No Units
AI19	Fault Hist 2	Fault History 2	P07.28	No Units
AI20	Fault Hist 3	Fault History 3	P07.29	No Units
AI21	Fault Hist 4	Fault History 4	P07.30	No Units
AI22	Fault Hist 5	Fault History 5	P07.31	No Units
AI23	Fault Hist 6	Fault History 6 (least recent)	P07.32	No Units

Table 29: Analog Output Object Instance Summary

Instance ID	Object Name	Description	Function Code	Units
AO1	Freq Cmd	Frequency command	P100.20	Hertz
AO2	PID Ref Cmd	PID reference command	P100.21	Percent
AO3	PID Fbk Cmd	PID feedback command	P100.22	Percent
AO4	Torque Cmd	Torque command	P100.23	Percent
AO5	Voltage Cmd	Voltage command	P100.24	Percent
AO6	Motor Group Sel	Motor group selection	P100.02	No Units

Table 30: Analog Value Object Instance Summary

Instance ID	Object Name	Description	Function Code	Units
AV1	Accel Time1	Acceleration time 1	P00.11	Seconds
AV2	Decel Time 1	Deceleration time 1	P00.12	Seconds

Table 31: Binary Input Object Instance Summary

Instance ID	Object Name	Description	Function Code	Bit	Active Text
					Inactive Text
BI1	Running/Stopped	Inverter running status	P100.51	0	Running
					Stopped
BI2	Reverse/Forward	Running direction	P100.51	1	Reverse
					Forward
BI3	Inverter Fault	Inverter fault present	P100.51	2	Fault
					OK
BI4	POFF	System power failure	P100.51	3	POFF
					OK
BI5	Bus Volt Present	DC bus voltage established	P100.51	8	Ready
					Not Rdy
BI6	Motor Type	Motor type feedback	P100.51	11	Synchron
					Inductn
BI7	Overload Warning	Overload pre-alarm	P100.51	12	Overload
					OK
BI8	Heartbeat Fbk	Heartbeat feedback value	P100.51	15	1
					0
BI9	Jogging	Inverter jogging status	P100.70	0	Jogging
					No Jog
BI10	FDT1	Frequency level detection FDT1	P100.70	1	Yes
					No
BI11	FDT2	Frequency level detection FDT2	P100.70	2	Yes
					No
BI12	Freq Arrival	Frequency arrival signal	P100.70	3	Yes
					No
BI13	Zero Speed	Running at zero speed	P100.70	4	Yes
					No
BI14	At Upper Freq	Upper limit frequency reached	P100.70	5	Yes
					No
BI15	At Lower Freq	Lower limit frequency reached	P100.70	6	Yes
					No
BI16	Pre-excitatn Fbk	Pre-excitation enable feedback	P100.70	7	Enabled
					Disabled
BI17	Undrload Warning	Underload pre-alarm	P100.70	8	Undrload
					OK
BI18	Speed Limiting	Frequency is limited	P100.70	14	Yes
					No
BI19	Control State	Toque/speed control state	P17.40	4	Torque
					Speed

Table 32: Binary Output Object Instance Summary

Instance ID	Object Name	Description	Function Code	Bit	Active Text
					Inactive Text
BO1	Run/Stop Cmd	Run/stop command	P100.01	0	Run
					Stop
BO2	Rev/Fwd Sel	Reverse/forward selection	P100.01	1	Reverse
					Forward
BO3	Jog Operation	Jog operation enable	P100.01	2	Enable
					Disable
BO4	Coast Stop	Coast to stop command	P100.01	3	Coast
					No Actn
BO5	Emergency Stop	Emergency stop command	P100.01	4	E Stop
					No Actn
BO6	Reset Fault	Reset fault command	P100.01	5	Reset
					No Actn
BO7	Ctrl Mode Switch	Control mode switching	P100.01	11	Enable
					Disable
BO8	Clear Pwr Cons	Clear power consumption	P100.01	12	Clear
					No Actn
BO9	Pre-excitation Cmd	Pre-excitation enable command	P100.01	13	Enable
					Disable
BO10	DC Brake Cmd	DC brake enable command	P100.01	14	Enable
					Disable
BO11	Heartbeat Cmd	Heartbeat reference command	P100.01	15	1
					0
BO12	Clear Fault Hist	Clear fault history	P00.18	1	Clear
					No Actn

10.1.3 Server Settings

In the studio's **Project** panel, navigate to **G500XLT...RS-485...BACnet MS/TP Server**.

Baud Rate

Selects the baud rate of the network.

Parity

Fixed at No Parity (1 Stop Bit).

APDU Timeout

Sets the time in milliseconds that the driver will wait for a response from a device after sending a request. As a BACnet server, this setting only applies to Confirmed COV Notifications. Note that this setting is only available for devices that support COV reporting.

Number of APDU Retries

Sets the number of times that the driver will retry a request when a response is not received. As a BACnet server, this setting only applies to Confirmed COV Notifications. Note that this setting is only available for devices that support COV reporting.

Max Master

Defines the highest allowable address (0...127) for MS/TP master nodes on the network. Any address higher than this will not receive the token from the driver. Note that this value must be greater than or equal to the driver's configured MAC Address(es). If the highest address on the network is unknown, set this field to 127.

Max Info Frames

Defines the maximum number of information frames the device may send before it must pass the token.

Fail-safe Timeout Time

Defines the maximum number of milliseconds for a break in network communications before a timeout event will be triggered. To disable timeout processing, set this field to 0.

Note that due to the nature of BACnet MS/TP, there is always token passing traffic which does not necessarily indicate the presence of a client. Therefore, "network communications" in the context of timeout processing is defined as the device receiving BACnet Confirmed Request PDUs from the client.

10.1.4 Node Settings

MAC Address

Defines the address (0...127) that the node will reside at on the network.

10.1.5 Device Object Settings

A Device Object is automatically added to the node, and cannot be removed. The Device Object contains several configurable fields that must be appropriately set for each device residing on a BACnet network.

Device Name

Defines the node's name. The device name must be unique across the entire BACnet network. Enter a string of between 1 and 16 characters in length.

Instance Number

Defines the node's instance number. The instance number must be unique across the entire BACnet network. Enter a value between 0...4194302 (0x0...0x3FFFFE).

10.1.6 BACnet Object Settings

In the studio's **Project** panel, navigate to **G500XLT...RS-485...BACnet MS/TP Server...Node** and add an object from the **Available Objects** panel.

The BACnet server hosts BACnet objects which contain many different properties for any BACnet client on the network to access. The driver supports a variety of different BACnet objects. All supported properties of these objects are readable, while the present value property is writable (for Outputs and Values only).

10.1.6.1 Analog Input Object Settings

Object Name

The name of the BACnet object. Enter a string of between 1 and 16 characters in length. All object names must be unique within a node.

Instance

The BACnet object's instance number. Enter a value between 0...4194302 (0x0...0x3FFFFE).

Function Code

The inverter function code (refer to section 5) that the BACnet object's present value will access.

Data Type

Specifies how the object's value will be stored internally in the device and whether the value should be treated as signed or unsigned.

Multiplier

The amount that associated network values are scaled by prior to being stored into the database or after being retrieved from the database. Upon retrieval from the database, raw data is multiplied by the multiplier to produce a network value (to send to a client).

Units

Select the desired units from this dropdown menu. If the desired units are not available in the dropdown menu, select "Other Units" and enter the appropriate enumerated value (as defined by the BACnet Specification) in the "Unit Value" field.

Unit Value

This field is enabled only when the "Units" selection is set to "Other Units". Enter the appropriate enumerated value (as defined by the BACnet Specification.)

Default COV Increment

Defines the default COV Increment value to use for this object for change of value detection. When COV reporting is enabled for this object, COV notifications will be generated if the present value property changes by the COV increment value, or more, since the last notification. A COV increment value of 0 indicates that any change to the present value property will generate a COV notification.

10.1.6.2 Analog Output Object Settings

Object Name

The name of the BACnet object. Enter a string of between 1 and 16 characters in length. All object names must be unique within a node.

Instance

The BACnet object's instance number. Enter a value between 0...4194302 (0x0...0x3FFFFE).

Function Code

The inverter function code (refer to section 5) that the BACnet object's present value will access.

Data Type

Specifies how the object's value will be stored internally in the device and whether the value should be treated as signed or unsigned.

Multiplier

The amount that associated network values are scaled by prior to being stored into the database or after being retrieved from the database. Upon retrieval from the database, raw data is multiplied by the multiplier to produce a network value (to send to a client).

Units

Select the desired units from this dropdown menu. If the desired units are not available in the dropdown menu, select "Other Units" and enter the appropriate enumerated value (as defined by the BACnet Specification) in the "Unit Value" field.

Unit Value

This field is enabled only when the "Units" selection is set to "Other Units". Enter the appropriate enumerated value (as defined by the BACnet Specification.)

Default COV Increment

Defines the default COV Increment value to use for this object for change of value detection. When COV reporting is enabled for this object, COV notifications will be generated if the present value property changes by the COV increment value, or more, since the last notification. A COV increment value of 0 indicates that any change to the present value property will generate a COV notification.

Relinquish Default

Defines the default value to be used for an object's present value property when all entries in the object's priority array are relinquished (written to NULL).

10.1.6.3 Analog Value Object Settings

Object Name

The name of the BACnet object. Enter a string of between 1 and 16 characters in length. All object names must be unique within a node.

Instance

The BACnet object's instance number. Enter a value between 0...4194302 (0x0...0x3FFFFE).

Function Code

The inverter function code (refer to section 5) that the BACnet object's present value will access.

Data Type

Specifies how the object's value will be stored internally in the device and whether the value should be treated as signed or unsigned.

Multiplier

The amount that associated network values are scaled by prior to being stored into the database or after being retrieved from the database. Upon retrieval from the database, raw data is multiplied by the multiplier to produce a network value (to send to a client).

Units

Select the desired units from this dropdown menu. If the desired units are not available in the dropdown menu, select "Other Units" and enter the appropriate enumerated value (as defined by the BACnet Specification) in the "Unit Value" field.

Unit Value

This field is enabled only when the "Units" selection is set to "Other Units". Enter the appropriate enumerated value (as defined by the BACnet Specification.)

Default COV Increment

Defines the default COV Increment value to use for this object for change of value detection. When COV reporting is enabled for this object, COV notifications will be generated if the present value property changes by the COV increment value, or more, since the last notification. A COV increment value of 0 indicates that any change to the present value property will generate a COV notification.

Relinquish Default

Defines the default value to be used for an object's present value property when all entries in the object's priority array are relinquished (written to NULL).

10.1.6.4 Binary Input Object Settings**Object Name**

The name of the BACnet object. Enter a string of between 1 and 16 characters in length. All object names must be unique within a node.

Instance

The BACnet object's instance number. Enter a value between 0...4194302 (0x0...0x3FFFFFFE).

Function Code

The inverter function code (refer to section 5) that the BACnet object's present value will access.

Bitmask

Specifies which bit(s) in the 16-bit value designated by the "Function Code" that the binary object will map to. This mechanism allows up to 16 binary objects to be simultaneously assigned to one register (each binary object mapping to a single bit of that 16-bit word). It is possible to map binary objects to multiple bits within the designated register.

The effect of the "Bitmask" field when reading: When the present value property of a binary object is read by a BACnet client, the bitmask is used to determine the active/inactive state of the object by inspecting the value in the designated register at the bit location(s) indicated in the bitmask. If all of the bit locations at the designated register are set, then the object's state will be returned as "active". Else, the object's state will be returned as "inactive". This resultant state is reversed prior to being placed on the network if the object's "Polarity" is set to "Reverse".

Active Text

Specifies the description of the object's "active" state. Enter a string of up to 8 characters in length. This field is optional and may be left blank.

Inactive Text

Specifies the description of the object's "inactive" state. Enter a string of up to 8 characters in length. This field is optional and may be left blank.

Polarity

Indicates the relationship between the physical state of the object (as stored in the register) and the logical state represented by the object's present value property. If the physical state is active high, select "Normal" from this dropdown menu. If the physical state is active low, select "Reverse" from this dropdown menu. For further detail, refer to the "Bitmask" behavioral description.

10.1.6.5 Binary Output Object Settings

Object Name

The name of the BACnet object. Enter a string of between 1 and 16 characters in length. All object names must be unique within a node.

Instance

The BACnet object's instance number. Enter a value between 0...4194302 (0x0...0x3FFFFE).

Function Code

The inverter function code (refer to section 5) that the BACnet object's present value will access.

Bitmask

Specifies which bit(s) in the 16-bit value designated by the "Function Code" that the binary object will map to. This mechanism allows up to 16 binary objects to be simultaneously assigned to one register (each binary object mapping to a single bit of that 16-bit word). It is possible to map binary objects to multiple bits within the designated register.

The effect of the "Bitmask" field when writing: When the present value property of a binary object is set to "active" by a BACnet client, then the bit(s) in the designated register indicated by the bitmask are set. Similarly, when the present value property of the object is set to "inactive", then the bit(s) in the designated register indicated by the bitmask are cleared. This setting/clearing behavior is reversed if the object's "Polarity" is set to "Reverse".

The effect of the "Bitmask" field when reading: When the present value property of a binary object is read by a BACnet client, the bitmask is used to determine the active/inactive state of the object by inspecting the value in the designated register at the bit location(s) indicated in the bitmask. If all of the bit locations at the designated register are set, then the object's state will be returned as "active". Else, the object's state will be returned as "inactive". This resultant state is reversed prior to being placed on the network if the object's "Polarity" is set to "Reverse".

Active Text

Specifies the description of the object's "active" state. Enter a string of up to 8 characters in length. This field is optional and may be left blank.

Inactive Text

Specifies the description of the object's "inactive" state. Enter a string of up to 8 characters in length. This field is optional and may be left blank.

Polarity

Indicates the relationship between the physical state of the object (as stored in the register) and the logical state represented by the object's present value property. If the physical state is active high, select "Normal" from this dropdown menu. If the physical state is active low, select "Reverse" from this dropdown menu. For further detail, refer to the "Bitmask" behavioral description.

Relinquish Default

Defines the default value to be used for an object's present value property when all entries in the object's priority array are relinquished (written to NULL).

10.1.6.6 Binary Value Object Settings

Object Name

The name of the BACnet object. Enter a string of between 1 and 16 characters in length. All object names must be unique within a node.

Instance

The BACnet object's instance number. Enter a value between 0...4194302 (0x0...0x3FFFFE).

Function Code

The inverter function code (refer to section 5) that the BACnet object's present value will access.

Bitmask

Specifies which bit(s) in the 16-bit value designated by the "Function Code" that the binary object will map to. This mechanism allows up to 16 binary objects to be simultaneously assigned to one register (each binary object mapping to a single bit of that 16-bit word). It is possible to map binary objects to multiple bits within the designated register.

The effect of the "Bitmask" field when writing: When the present value property of a binary object is set to "active" by a BACnet client, then the bit(s) in the designated register indicated by the bitmask are set. Similarly, when the present value property of the object is set to "inactive", then the bit(s) in the designated register indicated by the bitmask are cleared.

The effect of the "Bitmask" field when reading: When the present value property of a binary object is read by a BACnet client, the bitmask is used to determine the active/inactive state of the object by inspecting the value in the designated register at the bit location(s) indicated in the bitmask. If all of the bit locations at the designated register are set, then the object's state will be returned as "active". Else, the object's state will be returned as "inactive".

Active Text

Specifies the description of the object's "active" state. Enter a string of up to 8 characters in length. This field is optional and may be left blank.

Inactive Text

Specifies the description of the object's "inactive" state. Enter a string of up to 8 characters in length. This field is optional and may be left blank.

Relinquish Default

Defines the default value to be used for an object's present value property when all entries in the object's priority array are relinquished (written to NULL).

10.1.6.7 Multi-state Input Object Settings

Object Name

The name of the BACnet object. Enter a string of between 1 and 16 characters in length. All object names must be unique within a node.

Instance

The BACnet object's instance number. Enter a value between 0...4194302 (0x0...0x3FFFE).

Function Code

The inverter function code (refer to section 5) that the BACnet object's present value will access.

Data Type

Fixed at 16-Bit Unsigned.

Automatic Number of States

When enabled, this option automatically sets the Number of States setting to the largest value allowed by the selected Data Type.

Number of States

Defines the number of states that the object's present value property may have. The present value of a multi-state object is restricted to a range of 1...Number of States.

Offset by One

When enabled, this option automatically offsets the object's present value property by one compared to the object's value in the database. This option is useful when mapping a multi-state object to an enumeration that starts at zero, as zero is an invalid value for a multi-state object's present value property.

The effect of the "Offset by One" field when writing: When the present value property of a multi-state object is written by a BACnet client, the network value is decremented by one before being stored into the device's database.

The effect of the "Offset by One" field when reading: When the present value property of a multi-state object is read by a BACnet client, the raw data retrieved from the device's database is incremented by one to produce a network value.

10.1.6.8 Multi-state Output Object Settings

Object Name

The name of the BACnet object. Enter a string of between 1 and 16 characters in length. All object names must be unique within a node.

Instance

The BACnet object's instance number. Enter a value between 0...4194302 (0x0...0x3FFFFFFE).

Function Code

The inverter function code (refer to section 5) that the BACnet object's present value will access.

Data Type

Fixed at 16-Bit Unsigned.

Automatic Number of States

When enabled, this option automatically sets the Number of States setting to the largest value allowed by the selected Data Type.

Number of States

Defines the number of states that the object's present value property may have. The present value of a multi-state object is restricted to a range of 1...Number of States.

Offset by One

When enabled, this option automatically offsets the object's present value property by one compared to the object's value in the database. This option is useful when mapping a multi-state object to an enumeration that starts at zero, as zero is an invalid value for a multi-state object's present value property.

The effect of the "Offset by One" field when writing: When the present value property of a multi-state object is written by a BACnet client, the network value is decremented by one before being stored into the device's database.

The effect of the "Offset by One" field when reading: When the present value property of a multi-state object is read by a BACnet client, the raw data retrieved from the device's database is incremented by one to produce a network value.

Relinquish Default

Defines the default value to be used for an object's present value property when all entries in the object's priority array are relinquished (written to NULL).

10.1.6.9 Multi-state Value Object Settings

Object Name

The name of the BACnet object. Enter a string of between 1 and 16 characters in length. All object names must be unique within a node.

Instance

The BACnet object's instance number. Enter a value between 0...4194302 (0x0...0x3FFFFFFE).

Function Code

The inverter function code (refer to section 5) that the BACnet object's present value will access.

Data Type

Fixed at 16-Bit Unsigned.

Automatic Number of States

When enabled, this option automatically sets the Number of States setting to the largest value allowed by the selected Data Type.

Number of States

Defines the number of states that the object's present value property may have. The present value of a multi-state object is restricted to a range of 1...Number of States.

Offset by One

When enabled, this option automatically offsets the object's present value property by one compared to the object's value in the database. This option is useful when mapping a multi-state object to an enumeration that starts at zero, as zero is an invalid value for a multi-state object's present value property.

The effect of the "Offset by One" field when writing: When the present value property of a multi-state object is written by a BACnet client, the network value is decremented by one before being stored into the device's database.

The effect of the "Offset by One" field when reading: When the present value property of a multi-state object is read by a BACnet client, the raw data retrieved from the device's database is incremented by one to produce a network value.

Relinquish Default

Defines the default value to be used for an object's present value property when all entries in the object's priority array are relinquished (written to NULL).

10.2 Modbus RTU

10.2.1 Overview

The interface card supports the Modbus RTU protocol.

Other notes of interest include:

- Supported Modbus RTU functions are indicated in Table 33.
- The register mapping is provided in section 5.1.
- Inverter registers can be addressed as holding registers (4X references) and input registers (3X references).
- Specific bits within inverter registers can be accessed as either coils (0X references) or discrete inputs (1X references).
- Write data checking is not available (refer to section 5.2.) For example, if a write is performed to a register with a data value that is out-of-range of the corresponding parameter object, no Modbus exception will be immediately returned.
- The driver can be configured to detect a timeout (communication loss) and perform a timeout action.

10.2.2 Holding & Input Registers

By default, the inverter registers are mapped as both holding registers (4X) and input registers (3X) and are accessed by targeting the inverter register addresses (described in section 5.1) plus 1. Examples are shown in Table 34.

Table 34: Register Address-to-Register Number Examples

Function Code	Register Address	Conversion	Holding/Input Register Number
P17.01 (Output Frequency)	1701	1701 + 1	1702
P00.11 (Acceleration time 1)	11	11 + 1	12
P08.09 (Jump frequency 1)	809	809 + 1	810

The 4X and 3X only serve as a naming convention for holding register and input register respectively, and should NOT be included as part of the actual on-the-wire register number. To further clarify, Modbus register 41702 is the same as Modbus holding register 1702. The same description applies to input registers (3X).

For example, from a Modbus/TCP master's point of view, in order to access the output frequency (function code P17.01, register address 1701, register number 1702) as a holding register, the Modbus/TCP master must execute the Read Multiple Registers function code and target register 1702. This will similarly apply when accessing an inverter function code as an Input Register.

10.2.3 Coil & Discrete Input Mappings

The Modbus RTU driver provides read/write support for coils (0X references) and read-only support for discrete inputs (1X references). These will collectively be referred to from here on out as simply "discretes". Accessing discretes does not reference any new physical data: discretes are simply indexes into various bits of existing registers. What this means is that when a discrete is accessed, that discrete is resolved by the interface into a specific register, and a specific bit within that register. The pattern of discrete-to-register/bit relationships can be described as follows:

Discretes 1...16 map to register #1, bit0...bit15 (bit0=LSB, bit15=MSB)
 Discretes 17...32 map to register #2, bit0...bit15, and so on.

Arithmetically, the discrete-to-register/bit relationship can be described as follows: For any given discrete, the register in which that discrete resides can be determined by:

Table 33: Supported Modbus RTU Functions

Function Code	Function
1	Read coils
2	Read input status
3	Read multiple registers
4	Read input registers
5	Write coil
6	Write single register
8	Diagnostics (subfunction 0 only)
15	Force multiple coils
16	Write multiple registers

$$\text{register} = \left\lfloor \frac{\text{discrete} + 15}{16} \right\rfloor \quad \text{Equation 2}$$

Where the bracket symbols “ $\lfloor \rfloor$ ” indicate the “floor” function, which means that any fractional result (or “remainder”) is to be discarded, with only the integer value being retained.

Also, for any given discrete, the targeted bit in the register in which that discrete resides can be determined by:

$$\text{bit} = (\text{discrete} - 1) \% 16 \quad \text{Equation 3}$$

Where “discrete” $\in [1 \dots 65535]$, “bit” $\in [0 \dots 15]$, and “%” is the modulus operator, which means that any fractional result (or “remainder”) is to be retained, with the integer value being discarded (i.e. it is the opposite of the “floor” function).

For clarity, let’s use Equation 2 and Equation 3 in a calculation example. Say, for instance, that we are going to read coil #34. Using Equation 2, we can determine that coil #34 resides in register #3, as $\lfloor 3.0625 \rfloor = \lfloor 3 \text{ r}1 \rfloor = 3$. Then, using Equation 3, we can determine that the bit within register #3 that coil #34 targets is $(34-1)\%16 = 1$, as $33\%16 = \text{mod}(2 \text{ r}1) = 1$. Therefore, reading coil #34 will return the value of register #3, bit #1.

10.2.4 Slave Settings

Baud Rate

Selects the baud rate of the network.

Parity

Selects the parity and number of stop bits.

Timeout Time

Defines the maximum number of milliseconds for a break in network communications before a timeout event will be triggered. To disable timeout processing, set this field to 0.

Response Delay

Defines the time in milliseconds that the driver waits before responding to master requests. This is a useful feature for certain master devices or infrastructure components (such as radio modems) that may require a given amount of time to place themselves into a “receiving mode” where they are capable of listening for slave responses. If no delay is required, setting this field to 0 instructs the driver to send its responses as soon as possible.

10.2.5 Node Settings

Address

Defines the station address (1...247) for this node.

10.2.6 Holding/Input Register Remap Settings

In the studio’s **Project** panel, add **G500XLT...RS-485...Modbus RTU Slave...Holding/Input Register Remap**.

Holding/input register remap objects are **OPTIONAL**. By default, all inverter registers are already mapping as both holding (4X) and input (3X) registers (refer to section 10.2.2). For user convenience, register remap objects can be created to map any inverter register to holding/input register addresses 15000 to 15049 (i.e. holding/input register numbers 15001 to 15050).

At times, it may be convenient to access inverter function codes in bulk Modbus transactions. This may be especially true in situations where it is desired to access certain function codes that are non-contiguous. For example, if it were desired to read the inverter’s output frequency (function code P17.01, register number 1702), output current (function code P17.03, register number 1704), and output torque (function code P17.36, register number 1737), this could be accomplished in two different ways:

1. Implement three separate Modbus read transactions, each one reading one register only, or

2. Implement one single Modbus read transaction, starting at register number 1702 for a quantity of 36 registers. Then, pick out the registers of interest and ignore the rest of the response data.

While both of these methods will certainly work, neither one of them is optimized for the task at hand, which is to access three specific register values. A fully optimized solution can be realized by making use of the register remap objects. Non-contiguous inverter registers can be grouped together in any order and accessed efficiently via the Modbus RTU “read multiple registers” and “write multiple registers” function codes. The net effect is one of being able to transfer larger blocks of registers using fewer Modbus transactions, which results in improved network utilization and simpler data manipulation code on the Modbus master device.

Description

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

Remap Register Address

Remap register that maps to the specified inverter register. Select from 15000 to 15049.

Function Code

Function code (refer to section 5) that is accessed by the **Remap Register**.

10.3 Metasys N2

The interface card supports the Johnson Controls Metasys® N2 Open protocol. Some notes of interest are:

- Supports fully-configurable analog input, analog output, binary input, binary output, internal float, internal integer, and internal byte object types.
- The Metasys device type for the driver is VND.
- The driver can be configured to detect a timeout (communication loss) and perform a timeout action.

10.3.1 Default Supported Objects

This section will describe the default objects. Since the objects are configurable, the system integrator is responsible for managing, maintaining, and documenting the actual configuration.



Always use the studio to confirm the configuration before commissioning the device.

Table 35: Analog Input Object Instance Summary

Instance ID	Object Name	Description	Function Code	Units
AI1	Run Freq	Running frequency	P100.70	Hertz
AI2	Freq Ref	Frequency reference	P100.71	Hertz
AI3	Bus Voltage	DC bus voltage	P100.72	Volts
AI4	Output Voltage	Output voltage	P100.73	Volts
AI5	Output Current	Output current	P100.74	Amps
AI6	Output Torque	Actual output torque	P100.75	Percent
AI7	Output Power	Actual output power	P100.76	Percent
AI8	Fault Code	Fault code	P100.78	No Units
AI9	PID Ref	PID reference	P100.79	Percent
AI10	PID Fbk	PID feedback	P100.80	Percent
AI11	Torque Ref	Torque reference	P17.15	Percent
AI12	Operation Mode	Operation mode	P100.53	No Units
AI13	Motor Group Fbk	Motor group feedback	P100.52	No Units
AI14	Operation Time	Current running time	P17.26	Minutes
AI15	Mtr Power Factor	Motor power factor	P17.25	No Units
AI16	Power Cons Hi	High word of inverter power consumption	P07.15	Megawatt-hours
AI17	Power Cons Lo	Low word of inverter power consumption	P07.16	Kilowatt-hours
AI18	Fault Hist 1	Fault History 1 (most recent)	P07.27	No Units
AI19	Fault Hist 2	Fault History 2	P07.28	No Units
AI20	Fault Hist 3	Fault History 3	P07.29	No Units
AI21	Fault Hist 4	Fault History 4	P07.30	No Units
AI22	Fault Hist 5	Fault History 5	P07.31	No Units
AI23	Fault Hist 6	Fault History 6 (least recent)	P07.32	No Units

Table 36: Analog Output Object Instance Summary

Instance ID	Object Name	Description	Function Code	Units
AO1	Freq Cmd	Frequency command	P100.20	Hertz
AO2	PID Ref Cmd	PID reference command	P100.21	Percent
AO3	PID Fbk Cmd	PID feedback command	P100.22	Percent
AO4	Torque Cmd	Torque command	P100.23	Percent
AO5	Voltage Cmd	Voltage command	P100.24	Percent
AO6	Motor Group Sel	Motor group selection	P100.02	No Units
AO7	Accel Time1	Acceleration time 1	P00.11	Seconds
AO8	Decel Time 1	Deceleration time 1	P00.12	Seconds

Table 37: Binary Input Object Instance Summary

Instance ID	Object Name	Description	Function Code	Bit	1 State
					0 State
BI1	Running/Stopped	Inverter running status	P100.51	0	Running
					Stopped
BI2	Reverse/Forward	Running direction	P100.51	1	Reverse
					Forward
BI3	Inverter Fault	Inverter fault present	P100.51	2	Fault
					OK
BI4	POFF	System power failure	P100.51	3	POFF
					OK
BI5	Bus Volt Present	DC bus voltage established	P100.51	8	Ready
					Not Ready
BI6	Motor Type	Motor type feedback	P100.51	11	Synchronous
					Induction
BI7	Overload Warning	Overload pre-alarm	P100.51	12	Overload
					OK
BI8	Heartbeat Fbk	Heartbeat feedback value	P100.51	15	1
					0
BI9	Jogging	Inverter jogging status	P100.70	0	Jogging
					No Jog
BI10	FDT1	Frequency level detection FDT1	P100.70	1	Yes
					No
BI11	FDT2	Frequency level detection FDT2	P100.70	2	Yes
					No
BI12	Freq Arrival	Frequency arrival signal	P100.70	3	Yes
					No
BI13	Zero Speed	Running at zero speed	P100.70	4	Yes
					No
BI14	At Upper Freq	Upper limit frequency reached	P100.70	5	Yes
					No
BI15	At Lower Freq	Lower limit frequency reached	P100.70	6	Yes
					No
BI16	Pre-excitation Fbk	Pre-excitation enable feedback	P100.70	7	Enabled
					Disabled
BI17	Underload Warning	Underload pre-alarm	P100.70	8	Underload
					OK
BI18	Speed Limiting	Frequency is limited	P100.70	14	Yes
					No
BI19	Control State	Torque/speed control state	P17.40	4	Torque
					Speed

Table 38: Binary Output Object Instance Summary

Instance ID	Object Name	Description	Function Code	Bit	1 State
					0 State
BO1	Run/Stop Cmd	Run/stop command	P100.01	0	Run
					Stop
BO2	Rev/Fwd Sel	Reverse/forward selection	P100.01	1	Reverse
					Forward
BO3	Jog Operation	Jog operation enable	P100.01	2	Enable
					Disable
BO4	Coast Stop	Coast to stop command	P100.01	3	Coast
					No Action
BO5	Emergency Stop	Emergency stop command	P100.01	4	E Stop
					No Action
BO6	Reset Fault	Reset fault command	P100.01	5	Reset
					No Action
BO7	Ctrl Mode Switch	Control mode switching	P100.01	11	Enable
					Disable
BO8	Clear Pwr Cons	Clear power consumption	P100.01	12	Clear
					No Action
BO9	Pre-excitation Cmd	Pre-excitation enable command	P100.01	13	Enable
					Disable
BO10	DC Brake Cmd	DC brake enable command	P100.01	14	Enable
					Disable
BO11	Heartbeat Cmd	Heartbeat reference command	P100.01	15	1
					0
BO12	Clear Fault Hist	Clear fault history	P00.18	1	Clear
					No Action

10.3.2 Slave Settings

In the studio's **Project** panel, navigate to **G500XLT...RS-485...Metasys N2 Slave**.

Baud Rate

Fixed at 9600 baud.

Parity

Fixed at No Parity (1 Stop Bit).

Timeout Time

Defines the maximum number of milliseconds for a break in network communications before a timeout event will be triggered. To disable timeout processing, set this field to 0.

Response Delay

Defines the time (in milliseconds) that the driver waits before responding to master requests. This is a useful feature for certain master devices or infrastructure components (such as radio modems) that may require a given amount of time to place themselves into a "receiving mode" where they are capable of listening for slave responses. If no delay is required, setting this field to 0 instructs the driver to send its responses as soon as possible.

10.3.3 Node Settings

Address

Defines the station address (1...255) for this node.

10.3.4 Metasys Object Settings

In the studio's **Project** panel, navigate to **G500XLT...RS-485...Metasys N2 Slave...Node** and add an object from the **Available Objects** panel.

The N2 slave driver hosts Metasys objects which contain a variety of different attributes for an N2 master to access. The Metasys specification allows a maximum of 256 instances of each object type.

- **Analog input (AI)** objects are used for monitoring analog status items. AI objects support low alarm limits, low warning limits, high warning limits, high alarm limits and differential values. Change of state (COS), alarm and warning functions can also be enabled. An AI object will accept an override command, but will not change its actual value or indicate override active. A "multiplier" is associated with the object, which can provide scaling between the network value and raw (database) value.
- **Analog output (AO)** objects are used for setting and monitoring analog control and configuration items. An AO value can be modified by issuing an override command. Issuing a release command will not cause the AO to automatically return to its pre-override value, nor will the AO automatically return to its pre-override value after a certain time period of no communication. A "multiplier" is associated with the object, which can provide scaling between the network value and raw (database) value.
- **Binary input (BI)** objects are used for monitoring discrete (digital) status items. BI objects support COS, alarm enabling and normal/alarm status indications. A BI object will accept an override command, but will not change its actual value or indicate override active. A "bitmask" is associated with the object, which provides mapping information between the network value and specific bits in the database.
- **Binary output (BO)** points are used for setting and monitoring discrete control and configuration items. A BO value can be modified by issuing an override command. Issuing a release command will not cause the BO to automatically return to its pre-override value, nor will the BO return to its pre-override value after a certain time period of no communication. A "bitmask" is associated with the object, which provides mapping information between the network value and specific bits in the database.
- **Internal float (ADF)** objects are used for setting and monitoring internal floating point values. An internal float value can be modified by issuing an override command or a write attribute command. Issuing a release command will not cause the internal float to automatically return to its pre-override value, nor will the internal float automatically return to its pre-override value after a certain time period of no communication. A "multiplier" is associated with the object, which can provide scaling between the network value and raw (database) value.
- **Internal integer (ADI)** objects are used for setting and monitoring internal integer values. An internal integer value can be modified by issuing an override command or a write attribute command. Issuing a release command will not cause the internal integer to automatically return to its pre-override value, nor will the internal integer automatically return to its pre-override value after a certain time period of no communication. A "multiplier" is associated with the object, which can provide scaling between the network value and raw (database) value.
- **Internal byte (BD)** objects are used for setting and monitoring internal byte values. An internal byte value can be modified by issuing an override command or a write attribute command. Issuing a release command will not cause the internal byte to automatically return to its pre-override value, nor will the internal byte automatically return to its pre-override value after a certain time period of no communication. A "multiplier" is associated with the object, which can provide scaling between the network value and raw (database) value.

10.3.4.1 Analog Input Object Settings

Object Name

The name of the N2 object. Enter a string of between 1 and 32 characters in length.

Instance

The N2 object's instance number. Enter a value between 1...256.

Function Code

The inverter function code (refer to section 5) that the N2 object's value will access.

Data Type

Specifies how the object's value will be stored internally in the device and whether the value should be treated as signed or unsigned.

Multiplier

The amount that associated network values are scaled by prior to being stored into the database or after being retrieved from the database. Upon retrieval from the database, raw data is multiplied by the multiplier to produce a network value (to send to a master.)

10.3.4.2 Analog Output Object Settings

Object Name

The name of the N2 object. Enter a string of between 1 and 32 characters in length.

Instance

The N2 object's instance number. Enter a value between 1...256.

Function Code

The inverter function code (refer to section 5) that the N2 object's value will access.

Data Type

Specifies how the object's value will be stored internally in the device and whether the value should be treated as signed or unsigned.

Multiplier

The amount that associated network values are scaled by prior to being stored into the database or after being retrieved from the database. Upon retrieval from the database, raw data is multiplied by the multiplier to produce a network value (to send to a master.)

10.3.4.3 Binary Input Object Settings

Object Name

The name of the N2 object. Enter a string of between 1 and 32 characters in length.

Instance

The N2 object's instance number. Enter a value between 1...256.

Function Code

The inverter function code (refer to section 5) that the N2 object's value will access.

Bitmask

Specifies which bit(s) in the 16-bit value designated by the "Function Code" that the binary object will map to. This mechanism allows up to 16 binary objects to be simultaneously assigned to one register (each binary object mapping to a single bit of that 16-bit word). It is possible to map binary objects to multiple bits within the designated register.

The effect of the "Bitmask" field when reading: When the current state of a binary object is read by a Metasys master, the bitmask is used to determine the state of the object by inspecting the value in the designated register at the bit location(s) indicated in the bitmask. If all of the bit locations at the designated register indicated by a checkmark in the bitmask are set, then the object's state will be returned as "1". Else, the object's state will be returned as "0".

10.3.4.4 Binary Output Object Settings

Object Name

The name of the N2 object. Enter a string of between 1 and 32 characters in length.

Instance

The N2 object's instance number. Enter a value between 1...256.

Function Code

The inverter function code (refer to section 5) that the N2 object's value will access.

Bitmask

Specifies which bit(s) in the 16-bit value designated by the "Function Code" that the binary object will map to. This mechanism allows up to 16 binary objects to be simultaneously assigned to one register (each binary object mapping to a single bit of that 16-bit word). It is possible to map binary objects to multiple bits within the designated register.

The effect of the "Bitmask" field when reading: When the current state of a binary object is read by a Metasys master, the bitmask is used to determine the state of the object by inspecting the value in the designated register at the bit location(s) indicated in the bitmask. If all of the bit locations at the designated register indicated by a checkmark in the bitmask are set, then the object's state will be returned as "1". Else, the object's state will be returned as "0".

The effect of the "Bitmask" field when writing: When the current state of a binary object is overridden to "1" by a Metasys master, then the bit(s) in the designated register indicated by a checkmark in the bitmask are set. Similarly, when the current state of the object is overridden to "0", then the bit(s) in the designated register indicated by a checkmark in the bitmask are cleared.

10.3.4.5 Internal Float Object Settings**Object Name**

The name of the N2 object. Enter a string of between 1 and 32 characters in length.

Instance

The N2 object's instance number. Enter a value between 1...256.

Function Code

The inverter function code (refer to section 5) that the N2 object's value will access.

Data Type

Specifies how the object's value will be stored internally in the device and whether the value should be treated as signed or unsigned.

Multiplier

The amount that associated network values are scaled by prior to being stored into the database or after being retrieved from the database. Upon retrieval from the database, raw data is multiplied by the multiplier to produce a network value (to send to a master.)

10.3.4.6 Internal Integer Object Settings**Object Name**

The name of the N2 object. Enter a string of between 1 and 32 characters in length.

Instance

The N2 object's instance number. Enter a value between 1...256.

Function Code

The inverter function code (refer to section 5) that the N2 object's value will access.

Data Type

Specifies how the object's value will be stored internally in the device and whether the value should be treated as signed or unsigned.

Multiplier

The amount that associated network values are scaled by prior to being stored into the database or after being retrieved from the database. Upon retrieval from the database, raw data is multiplied by the multiplier to produce a network value (to send to a master.)

10.3.4.7 Internal Byte Object Settings**Object Name**

The name of the N2 object. Enter a string of between 1 and 32 characters in length.

Instance

The N2 object's instance number. Enter a value between 1...256.

Function Code

The inverter function code (refer to section 5) that the N2 object's value will access.

Data Type

Specifies how the object's value will be stored internally in the device and whether the value should be treated as signed or unsigned.

Multiplier

The amount that associated network values are scaled by prior to being stored into the database or after being retrieved from the database. Upon retrieval from the database, raw data is multiplied by the multiplier to produce a network value (to send to a master.)

10.4 Siemens FLN (P1)

The interface card supports the Siemens FLN (P1) protocol. Some notes of interest are:

- Supports fully-configurable logical analog input (LAI), logical analog output (LAO), logical digital input (LDI), logical digital output (LDO) point types.
- Supports custom application number, application descriptor, revision string, and revision number.
- The driver can be configured to detect a timeout (communication loss) and perform a timeout action.

10.4.1 Default Supported Objects

This section will describe the default objects. Since the objects are configurable, the system integrator is responsible for managing, maintaining, and documenting the actual configuration.



Always use the studio to confirm the configuration before commissioning the device.

Table 39: LAI Object Summary

Point Number	Object Name	Description	Function Code	Slope	Intercept	Units
3	RUN FREQ	Running frequency	P100.70	0.01	0	HZ
4	FREQ REF	Frequency reference	P100.71	0.01	0	HZ
5	BUS VOLTAGE	DC bus voltage	P100.72	0.1	0	V
6	OUTPUT VOLTS	Output voltage	P100.73	1	0	V
7	OUTPUT CURR	Output current	P100.74	0.1	0	A
8	OUTPUT TORQ	Actual output torque	P100.75	0.1	0	PCT
9	OUTPUT PWR	Actual output power	P100.76	0.1	0	PCT
10	FAULT CODE	Fault code	P100.78	1	0	
11	PID REF	PID reference	P100.79	0.1	0	PCT
12	PID FBK	PID feedback	P100.80	0.1	0	PCT
13	TORQUE REF	Torque reference	P17.15	0.1	0	PCT
14	OP MODE	Operation mode	P100.53	1	0	
15	MTR GRP FBK	Motor group feedback	P100.52	1	0	
16	OP TIME	Current running time	P17.26	1	0	MINS
17	MTR PWR FCTR	Motor power factor	P17.25	0.01	0	
18	PWR CONS HI	High word of inverter power consumption	P07.15	1	0	MWH
19	PWR CONS LO	Low word of inverter power consumption	P07.16	1	0	KWH
21	FAULT HIST 1	Fault History 1 (most recent)	P07.27	1	0	
22	FAULT HIST 2	Fault History 2	P07.28	1	0	
23	FAULT HIST 3	Fault History 3	P07.29	1	0	
24	FAULT HIST 4	Fault History 4	P07.30	1	0	
25	FAULT HIST 5	Fault History 5	P07.31	1	0	
26	FAULT HIST 6	Fault History 6 (least recent)	P07.32	1	0	

Table 40: LAO Object Summary

Point Number	Object Name	Description	Function Code	Slope	Intercept	Units
30	FREQ CMD	Frequency command	P100.20	0.01	0	HZ
31	PID REF CMD	PID reference command	P100.21	0.1	0	PCT
32	PID FBK CMD	PID feedback command	P100.22	0.1	0	PCT
33	TORQUE CMD	Torque command	P100.23	0.1	0	PCT
34	VOLTAGE CMD	Voltage command	P100.24	0.1	0	PCT
35	MTR GRP SEL	Motor group selection	P100.02	1	0	
36	ACCEL TIME 1	Acceleration time 1	P00.11	0.1	0	SECS
37	DECEL TIME 1	Deceleration time 1	P00.12	0.1	0	SECS

Table 41: LDI Object Summary

Point Number	Object Name	Description	Function Code	Bit	On Text
					Off Text
38	RUN.STOP	Inverter running status	P100.51	0	RUN
					STOP
39	REV.FWD	Running direction	P100.51	1	REV
					FWD
40	INVTR FAULT	Inverter fault present	P100.51	2	FAULT
					OK
41	POFF	System power failure	P100.51	3	POFF
					OK
42	BUS VOLT	DC bus voltage established	P100.51	8	READY
					NOTRDY
43	MOTOR TYPE	Motor type feedback	P100.51	11	SYNCHR
					INDUCT
44	OVRLD WARN	Overload pre-alarm	P100.51	12	OVRLD
					OK
45	HEARTBT FBK	Heartbeat feedback value	P100.51	15	1
					0
46	JOGGING	Inverter jogging status	P100.70	0	JOG
					NO JOG
47	FDT1	Frequency level detection FDT1	P100.70	1	YES
					NO
48	FDT2	Frequency level detection FDT2	P100.70	2	YES
					NO
49	FREQ ARRIVAL	Frequency arrival signal	P100.70	3	YES
					NO
50	ZERO SPEED	Running at zero speed	P100.70	4	YES
					NO
51	AT UPPR FREQ	Upper limit frequency reached	P100.70	5	YES
					NO
52	AT LOWR FREQ	Lower limit frequency reached	P100.70	6	YES
					NO
53	PREEXCIT FBK	Pre-excitation enable feedback	P100.70	7	EN
					DIS
54	UNDRLD WARN	Underload pre-alarm	P100.70	8	UNDRLD
					OK
55	SPEED LIMIT	Frequency is limited	P100.70	14	YES
					NO
56	CTRL STATE	Torque/speed control state	P17.40	4	TORQUE
					SPEED

Table 42: LDO Object Summary

Point Number	Object Name	Description	Function Code	Bit	On Text
					Off Text
57	RUN.STOP CMD	Run/stop command	P100.01	0	RUN
					STOP
58	REV.FWD SEL	Reverse/forward selection	P100.01	1	REV
					FWD
59	JOG OP	Jog operation enable	P100.01	2	EN
					DIS
60	COAST STOP	Coast to stop command	P100.01	3	COAST
					NO ACT
61	EMRGNCY STOP	Emergency stop command	P100.01	4	E STOP
					NO ACT
62	RESET FAULT	Reset fault command	P100.01	5	RESET
					NO ACT
63	CTRL MODE SW	Control mode switching	P100.01	11	EN
					DIS
64	CLR PWR CONS	Clear power consumption	P100.01	12	CLEAR
					NO ACT
65	PREEXCITN	Pre-excitation enable	P100.01	13	EN
					DIS
66	DC BRAKE	DC brake enable	P100.01	14	EN
					DIS
67	HEARTBT CMD	Heartbeat reference command	P100.01	15	1
					0
68	CLR FLT HIST	Clear fault history	P00.18	1	CLEAR
					NO ACT

10.4.2 Slave Settings

In the studio's **Project** panel, navigate to **G500XLT...RS-485...Siemens FLN Slave**.

Baud Rate

Selects the baud rate of the network.

Parity

Fixed at No Parity (1 Stop Bit).

Timeout Time

Defines the maximum number of milliseconds for a break in network communications before a timeout event will be triggered. To disable timeout processing, set this field to 0.

Response Delay

Defines the time (in milliseconds) that the driver waits before responding to controller requests. This is a useful feature for certain controllers or infrastructure components (such as radio modems) that may require a given amount of time to place themselves into a "receiving mode" where they are capable of listening for slave responses. If no delay is required, setting this field to 0 instructs the driver to send its responses as soon as possible.

10.4.3 Node Settings

In the studio's **Project** panel, navigate to **G500XLT...RS-485...Siemens FLN...Node**.

Address

Defines the station address (0...98) for this node.

Application Number

Defines the unique application number for the application. The default is 18260.

Application Descriptor

Enter the application descriptor string (up to 12 ASCII characters). The default is "GALT G500".

Revision String

Defines the unique 4-character revision string. The default is MP10.

Revision Number

Defines the revision number (0...255) for this application. The default is 1.

10.4.4 FLN Object Settings

In the studio's **Project** panel, navigate to **G500XLT...RS-485...Siemens FLN...Node** and add an object from the **Available Objects** panel.

10.4.4.1 Logical Analog Input (LAI) Object Settings**Point Descriptor**

The name of the point. Enter a string of between 1 and 12 characters in length. All point names must be unique within a node, and characters must be valid for encoding in RAD50 format ("A"... "Z", "0"... "9", blank, "\$", ".", and "?").

Point Number

The point number (1...99). Note that point numbers that are reserved by the FLN specification cannot be used (1, 2, 20, 29 and 99).

Function Code

The inverter function code (refer to section 5) that the point's value will access.

Data Type

Fixed at 16-Bit Unsigned.

Default Value

Defines the factory default physical value of the point (0...32767).

Max Value

Defines the maximum physical value that this point can attain (0...32767).

Slope

Defines the floating-point slope value of the point. The master uses this value to calculate a floating-point analog value using the equation $y=mx+b$, where y is the analog value, x is the physical value stored in the database, m is the slope, and b is the intercept.

Intercept

Defines the floating-point intercept value of the point. The master uses this value to calculate a floating-point analog value using the equation $y=mx+b$, where y is the analog value, x is the physical value stored in the database, m is the slope, and b is the intercept.

Units

Enter an ASCII string of up to 6 characters in length which represent the engineering units of the point. Characters must be valid for encoding in RAD50 format ("A"... "Z", "0"... "9", blank, "\$", ".", and "?").

10.4.4.2 Logical Analog Output (LAO) Object Settings**Point Descriptor**

The name of the point. Enter a string of between 1 and 12 characters in length. All point names must be unique within a node, and characters must be valid for encoding in RAD50 format ("A"... "Z", "0"... "9", blank, "\$", ".", and "?").

Point Number

The point number (1...99). Note that point numbers that are reserved by the FLN specification cannot be used (1, 2, 20, 29 and 99).

Function Code

The inverter function code (refer to section 5) that the point's value will access.

Data Type

Fixed at 16-Bit Unsigned.

Default Value

Defines the factory default physical value of the point (0...32767).

Max Value

Defines the maximum physical value that this point can attain (0...32767).

Slope

Defines the floating-point slope value of the point. The master uses this value to calculate an analog value using the equation $y = mx + b$, where y is the analog value, x is the physical value, m is the slope, and b is the intercept.

Intercept

Defines the floating-point intercept value of the point. The master uses this value to calculate an analog value using the equation $y = mx + b$, where y is the analog value, x is the physical value, m is the slope, and b is the intercept.

Units

Enter an ASCII string of up to 6 characters in length which represent the engineering units of the point. Characters must be valid for encoding in RAD50 format ("A"... "Z", "0"... "9", blank, "\$", ".", and "?").

10.4.4.3 Logical Digital Input (LDI) Object Settings

Point Descriptor

The name of the point. Enter a string of between 1 and 12 characters in length. All point names must be unique within a node, and characters must be valid for encoding in RAD50 format ("A"... "Z", "0"... "9", blank, "\$", ".", and "?").

Point Number

The point number (1...99). Note that point numbers that are reserved by the FLN specification cannot be used (1, 2, 20, 29 and 99).

Function Code

The inverter function code (refer to section 5) that the point's value will access.

Bitmask

Specifies which bit(s) in the 16-bit value designated by the "Function Code" that the binary object will map to. This mechanism allows up to 16 LDI objects to be simultaneously assigned to one register (each LDI object mapping to a single bit of that 16-bit word). It is possible to map LDI objects to multiple bits within the designated register.

The effect of the "Bitmask" field when reading: When the value of an LDI object is read by an FLN controller, the bitmask is used to determine the value of the object by inspecting the value in the designated register at the bit location(s) indicated in the bitmask. If all of the bit locations at the designated register indicated by a checkmark in the bitmask are set, then the object's value will be returned as "on". Else, the object's value will be returned as "off".

On Text

Enter an ASCII string of up to 6 characters in length which represent the On Text of the point. Characters must be valid for encoding in RAD50 format ("A"... "Z", "0"... "9", blank, "\$", ".", and "?").

Off Text

Enter an ASCII string of up to 6 characters in length which represent the Off Text of the point. Characters must be valid for encoding in RAD50 format ("A"... "Z", "0"... "9", blank, "\$", ".", and "?").

10.4.4.4 Logical Digital Output (LDO) Object Settings

Point Descriptor

The name of the point. Enter a string of between 1 and 12 characters in length. All point names must be unique within a node, and characters must be valid for encoding in RAD50 format ("A"... "Z", "0"... "9", blank, "\$", ".", and "?").

Point Number

The point number (1...99). Note that point numbers that are reserved by the FLN specification cannot be used (1, 2, 20, 29 and 99).

Function Code

The inverter function code (refer to section 5) that the point's value will access.

Bitmask

Specifies which bit(s) in the 16-bit value designated by the "Function Code" that the binary object will map to. This mechanism allows up to 16 LDI objects to be simultaneously assigned to one register (each LDI object mapping to a single bit of that 16-bit word). It is possible to map LDI objects to multiple bits within the designated register.

The effect of the "Bitmask" field when reading: When the value of an LDI object is read by an FLN controller, the bitmask is used to determine the value of the object by inspecting the value in the designated register at the bit location(s) indicated in the bitmask. If **all** of the bit locations at the designated register indicated by a checkmark in the bitmask are set, then the object's value will be returned as "on". Else, the object's value will be returned as "off".

The effect of the "Bitmask" field when writing: When the value of an LDO object is commanded to "on" by an FLN controller, then the bit(s) in the designated register indicated by a checkmark in the bitmask are set. Similarly, when the value of an LDO object is commanded to "off", then the bit(s) in the designated register indicated by a checkmark in the bitmask are cleared.

On Text

Enter an ASCII string of up to 6 characters in length which represent the On Text of the point. Characters must be valid for encoding in RAD50 format ("A"... "Z", "0"... "9", blank, "\$", ".", and "?").

Off Text

Enter an ASCII string of up to 6 characters in length which represent the Off Text of the point. Characters must be valid for encoding in RAD50 format ("A"... "Z", "0"... "9", blank, "\$", ".", and "?").

10.5 DMX-512

This interface card supports the DMX-512 protocol, which allows any connected non-DMX equipment to receive data from a universal DMX controller device. Some notes of interest are:

- Fully configurable to occupy any number of sequential DMX channels.
- Capable of using all 512 DMX channels.
- Each inverter parameter uses 2 DMX channels with configurable byte order.
- The driver can be configured to detect a timeout (communication loss) and perform a timeout action.

10.5.1 Connections

This section describes the typical connections used to connect a DMX controller device to the interface card.

While there are a variety of different DMX-512 connector types in existence, most standard DMX-512 connectors use either XLR 5-pin or 3-pin connectors (refer to Figure 14 and Figure 15). A female connector is fitted to a transmitter device (e.g. a console,) while a male connector is fitted to a receiver device (e.g. a dimmer or servo).



An



Figure 14: 5-Pin XLR Connector

Figure 15: 3-Pin XLR Connectors

appropriate wiring harness must be used when connecting the DMX-512 network to the interface card's RS-485 port. This can be accomplished by using off-the-shelf DMX-512 cabling with bare-wire terminations on one end, or by simply cutting a standard DMX-512 cable in half and stripping back the wires. Refer to Table 43 for an overview of DMX-512 pin assignments and connections.

Table 43: DMX-512 Pin Assignments

Pin	Usage	Card Connection
1	Network GND reference	COM
2	Primary data-	-
3	Primary data+	+
4	Optional secondary data- (not available on 3-pin connectors)	N/A
5	Optional secondary data+ (not available on 3-pin connectors)	N/A

10.5.2 Default Supported Objects

This section will describe the default objects. Since the objects are configurable, the system integrator is responsible for managing, maintaining, and documenting the actual configuration.



Always use the studio to confirm the configuration before commissioning the device.

Table 44: DMX-512 Channel Summary

Channel	Name	Description	Function Code (Bit)	Resolution	Off Value	On Value
1	Run/Stop Cmd	Run/stop command	P100.01 (0)	8-Bit	0...127	128...255
2	Rev/Fwd Cmd	Reverse/forward selection	P100.01 (1)	8-Bit	0...127	128...255
3	Freq Cmd	Frequency command (coarse)	P100.20	16-Bit	N/A	N/A
4		Frequency command (fine)				

10.5.3 Slave Settings

In the studio's **Project** panel, navigate to **G500XLT...RS-485...DMX-512 Slave**.

Baud Rate

Fixed at 250kbaud.

Parity

Fixed at No Parity (2 Stop Bit).

Timeout Time

Defines the maximum number of milliseconds for a break in network communications before a timeout event will be triggered. To disable timeout processing, set this field to 0.

10.5.4 Node Settings

Address

Defines the DMX channel number start address (1...512) for this node.

10.5.5 DMX Parameter Settings

In the studio's **Project** panel, navigate to **G500XLT...RS-485...DMX-512 Slave...Node** and add an object from the **Available Objects** panel.

Description

The description of the parameter. Enter a string of between 1 and 32 characters in length.

Function Code

The inverter function code (refer to section 5) that the parameter's value will access.

Resolution

Fixed at 16-Bit.

Channel Order

Selects the order of the channels that comprise the value of this parameter. High byte first means the first DMX channel is the most significant byte, or coarse value. Low byte first means the first DMX channel is the least significant byte, or fine value.

10.6 Generic Serial

The Generic Serial driver can be used to communicate as a slave with any serial device using configurable ASCII and/or binary data packets. This includes devices such as ASCII serial devices and devices using custom or proprietary serial protocols. Some notes of interest are:

- Supports communication with almost any serial device.
- Supports both master-slave transactions and producer-consumer transactions.
- Transactions are defined at a packet level by adding configurable packet data objects.
- Versatile packet matching options allow handling a variety of different packets with a single transaction definition.
- Requests are matched to transactions based on the order that the transactions are defined. This allows defining a "default" or "exception" transaction last to be initiated for undefined requests.
- Transactions support using database logic to manipulate received data before placing the result into the response.
- Supports variable-sized data fields and packets containing a variable number of data elements.
- Supports binary, ASCII hexadecimal, ASCII decimal, and ASCII text data encodings.
- Supports unsigned integer, signed integer, and IEEE-754 floating point number formats.
- Full support for 8-bit, 16-bit, and 32-bit checksum and CRC fields.
- The driver can be configured to detect a timeout (communication loss) and perform a timeout action.

10.6.1 Slave Settings

In the studio's **Project** panel, navigate to **G500XLT...RS-485...Generic Serial Slave**.

Data Bits

Select between 7 or 8 data bits per character.

Baud Rate

Selects the baud rate of the network.

Parity

Selects the parity and number of stop bits.

Packet Gap Interval

Defines the number of character times of silence on the network that indicates the end of a packet.

Timeout

Defines the maximum number of milliseconds for a break in network communications before a timeout event will be triggered. To disable timeout processing, set this field to 0.

Response Delay

Defines the time in milliseconds that the driver waits before responding to master requests. This is a useful feature for certain master devices or infrastructure components (such as radio modems) that may require a given amount of time to place themselves into a "receiving mode" where they are capable of listening for slave responses. If no delay is required, setting this field to 0 instructs the driver to send its responses as soon as possible.

10.6.2 Transactions

Communications is established by defining configurable transactions which will be matched to a received packet based on the order they are defined. Each transaction consists of a packet expected to be received from a device on the network and, optionally, a response packet to transmit back to that device.

10.6.2.1 Slave Transaction

In the studio's **Project** panel, navigate to **G500XLT...RS-485...Generic Serial Slave...Slave Transaction** and add an object from the **Available Objects** panel to the **Expected Request** or **Response** packet definition.

A slave transaction is a transaction in which a master device sends a request to the driver and expects a response. When a slave transaction is added to the configuration, an expected request and response is automatically added below it. The expected request defines a packet that is expected to be received from the master. The response defines the packet to send to the master after receiving the request.

10.6.2.2 Consumer Transaction

In the studio's **Project** panel, navigate to **G500XLT...RS-485...Generic Serial Slave...Consumer Transaction** and add an object from the **Available Objects** panel to the **Expected Consumed Data** packet definition.

A consumer transaction is a transaction in which a producer device sends a data packet to the driver and does not expect a response. When a consumer transaction is added to the configuration, a consumed data packet is automatically added below it. The consumed data packet defines a packet that is expected to be received from producers on the network.

10.6.2.3 Transaction Settings

Description

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

Uses Database Logic

Check this option if the transaction requires database logic to run between receiving the expected request and matching the response.

10.6.2.4 Received Event

Each transaction can optionally include a received event. This adds the ability to detect when a packet matching the associated transaction has been received. The event is assigned to a register in the database. When a packet matching the transaction has been received, the event database location is set to a value of 1.

Received Event Function Code

Specifies the function code to use for detecting that a packet matching the associated transaction has been received.

10.6.3 Packet Data Objects

Packet data objects are used to build a packet. Every character or field that is present in a packet is defined by one or more packet data objects. The order of the fields in the packet dictates the order that the corresponding packet data objects must be added. There are various types of packet data objects available in order to facilitate the definition of a packet.

10.6.3.1 Constant Data

Adds constant data characters to the packet.

Name

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

Constant Data Characters

Defines the constant data that is encoded in the packet. Enter up to 16 hexadecimal bytes or ASCII characters.

10.6.3.2 Drive Register Data

Adds data that is mapped to the inverter's registers. For transmitted packets, the data is read from the inverter and put into the packet. For received packets, the data from the packet is written to the inverter.

Name

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

Element Encoding

Selects the encoding used for a data element in the packet.

Number Format

Selects the format for interpreting the value of a number.

Fixed Element Size

Check this option if the size of a data element is fixed. If this option is unchecked, the driver will automatically determine the size of a variable-sized data element.

Element Size

Enter the number of characters reserved for each data element in the packet.

Byte Order

Selects the byte ordering used for multi-byte data elements in the packet.

Number of Elements

Defines the number of data elements in the packet to map to the device's database.

Starting Function Code

Defines the starting inverter function code where the packet's data elements are mapped.

Internal Data Type

Specifies how each data element in the packet will be stored internally in the option card. This defines the resolution of the value and whether the value should be treated as signed or unsigned.

Multiplier

The amount that data values are scaled by prior to being stored into the database or after being retrieved from the database. Prior to storage into the database, data values are divided by the multiplier to produce database values. Upon retrieval from the database, database values are multiplied by the multiplier to produce data values.

10.6.3.3 *Variable Drive Register Data*

Adds data that is mapped to the device's database which consists of a variable number of elements. The data from a received packet is written to the database. If the actual number of elements in the packet is less than the defined maximum, the remaining database values will be set to 0.

Name

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

Element Encoding

Selects the encoding used for a data element in the packet.

Number Format

Selects the format for interpreting the value of a number.

Fixed Element Size

Check this option if the size of a data element is fixed. If this option is unchecked, the driver will automatically determine the size of a variable-sized data element.

Element Size

Enter the number of characters reserved for each data element in the packet.

Byte Order

Selects the byte ordering used for multi-byte data elements in the packet.

Max Number of Elements

Defines the maximum number of data elements in the packet to map to the device's database. If the actual number of elements in the packet is less than this, the remaining database values will be 0.

Starting Function Code

Defines the starting inverter function code where the packet's data elements are mapped.

Internal Data Type

Specifies how each data element in the packet will be stored internally in the option card. This defines the resolution of the value and whether the value should be treated as signed or unsigned.

Multiplier

The amount that data values are scaled by prior to being stored into the database or after being retrieved from the database. Prior to storage into the database, data values are divided by the multiplier

to produce database values. Upon retrieval from the database, database values are multiplied by the multiplier to produce data values.

10.6.3.4 Ignored Characters

Adds "Don't Care" characters whose values are ignored.

Name

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

Number of Characters

Defines the number of "Don't Care" characters in the packet that will be ignored.

10.6.3.5 Variable Ignored Characters

Adds a variable number of "Don't Care" characters whose values are ignored.

Name

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

Max Number of Characters

Defines the maximum number of "Don't Care" characters in the packet that will be ignored.

10.6.3.6 Ranged Byte

Adds a single byte that must be within a defined range.

Name

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

Min Value

Defines the minimum value the byte can have to be considered a match.

Max Value

Defines the maximum value the byte can have to be considered a match.

10.6.3.7 Bitmasked Byte

Adds a single byte that must match a defined value after a bitmask is applied.

Name

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

Bitmask

Specifies the bit(s) in the byte that are relevant for detecting a match.

Value

Defines the value the byte must have to be considered a match after applying the bitmask.

10.6.3.8 Register Matched Byte

Adds a single byte that must match the current value stored at a location in one of the drive's registers.

Name

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

Starting Function Code

Defines the inverter function code containing the value the byte must have to be considered a match.

10.6.3.9 List Matched Character

Adds a single character that must match a value from a list of values.

Name

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

Value List

Defines a list of possible values the character can have to be considered a match. Enter up to 15 hexadecimal bytes or ASCII characters.

10.6.3.10 Checksum

Adds a checksum field that is calculated from a defined start offset in the packet up to, but not including, the location of the checksum field itself.

Name

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

Checksum Data Type

Selects the data type and defines the width of the checksum.

Encoding

Selects the encoding used for the checksum in the packet.

Checksum Size

The number of characters reserved for the checksum in the packet.

Byte Order

Selects the byte ordering for how the checksum is encoded in the packet.

Checksum Calculation Parameters

Start Offset

Defines the starting offset in the packet where the checksum calculation will begin.

Use 2 Chars/Byte

Check this option to first convert 2 ASCII characters from the packet into a byte value which will be used in the checksum calculation. If this option is unchecked, the raw bytes from the packet will be used directly in the checksum calculation.

Final Operation

Selects an optional final operation to apply to the checksum value at the end of the calculation.

10.6.3.11 CRC

Adds a cyclic redundancy check field that is calculated from a defined start offset in the packet up to, but not including, the location of the CRC field itself.

Name

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

CRC Data Type

Selects the data type and defines the width of the CRC.

Encoding

Selects the encoding used for the CRC in the packet.

CRC Size

The number of characters reserved for the CRC in the packet.

Byte Order

Selects the byte ordering for how the CRC is encoded in the packet.

CRC Calculation Parameters

Start Offset

Defines the starting offset in the packet where the CRC calculation will begin.

Use 2 Chars/Byte

Check this option to first convert 2 ASCII characters from the packet into a byte value which will be used in the CRC calculation. If this option is unchecked, the raw bytes from the packet will be used directly in the CRC calculation.

Bit Order (Shift Direction)

Selects which bit in each byte is considered first (and by correlation, the direction bits are shifted) when calculating the CRC.

Polynomial

Defines the generator polynomial used in the CRC calculation. Enter the hexadecimal representation of the polynomial's bit sequence, most-significant bit first, omitting the highest-order bit.

Initial Value

Enter the value that the CRC will be initialized to at the beginning of the CRC calculation.

Final XOR Value

Enter the value that the CRC will be XORed with at the end of the CRC calculation.

11 TROUBLESHOOTING

Although by no means exhaustive, Table 45 provides possible causes behind some of the most common errors experienced when using the interface card.

Table 45: Troubleshooting

Problem	Symptom	Solution
Inverter fault	Inverter displays a fault code	<ul style="list-style-type: none"> Refer to the Troubleshooting section in the G500 Operation Manual.
No communications between the network and the card	Communications cannot be established, the RS-485 "TX" LED is off, or the RS-485 "RX" LED flashes only infrequently, not at all, or is solid on	<ul style="list-style-type: none"> Confirm that the card is running normally (Module Status LED is solid green, not blinking red or rapidly blinking green). Refer to section 1.4.1. Confirm that the RS-485 wiring is correct as described in section 3. Ensure that the card is programmed with compatible network settings. Refer to section 6. If the "RX" LED is solid on, this typically indicates reversed wiring polarity. Swap the RS-485 + and - wires. If the "RX" LED is off, this indicates the card is not receiving any data. Confirm that the master device is sending requests on the network.
Unable to control the inverter via network communications	Writing to command and frequency function codes/registers has no apparent effect on inverter operation	<ul style="list-style-type: none"> Confirm that the applicable inverter function codes are set to allow network control (refer to section 4). If using the inverter's terminal contacts, refer to the inverter's instruction manual to determine the appropriate behavior and priority.
NPU or studio cannot discover the card	The NPU does not connect or the studio does not display the card under "Online Devices"	<ul style="list-style-type: none"> Confirm that the card is running normally and connected via USB. Confirm that the status LED blinks the green/red startup sequence when power is first applied. Confirm that the status LED blinks green while a USB cable is connected. Reinstall the Configuration Studio and run the USB driver installer again.
Firmware-generated error	"MODULE STATUS" LED is flashing red. The number of times the LED flashes indicates an error code.	<ul style="list-style-type: none"> 6 flashes indicates the card is in USB to Serial Pass-Through mode. 7 flashes indicates invalid configuration parameters. Ensure that the configuration loaded on the option card using the Configuration Studio contains a definition for the selected protocol set via the NPU. 9 flashes indicates a host communication error. Confirm that the inverter is powered and the card is properly seated. Record the error code blinking pattern and contact technical support for further assistance.



GALT ELECTRIC

California USA
www.galtelectric.com
1-800-511-7734