INDUSTRIAL CONTROL COMMUNICATIONS, INC.

PicoPort Datasheet

BACnet is a registered trademark of ASHRAE. ASHRAE does not endorse, approve or test products for compliance with ASHRAE standards. Compliance of listed products to the requirements of ASHRAE Standard 135 is the responsibility of BACnet International (BI). BTL is a registered trademark of BI.

December 12, 2016

PicoPort Datasheet

NOTICE TO USERS

Industrial Control Communications, Inc. reserves the right to make changes and improvements to its products without providing notice.

Industrial Control Communications, Inc. shall not be liable for technical or editorial omissions or mistakes in this datasheet, nor shall it be liable for incidental or consequential damages resulting from the use of information contained in this datasheet.

INDUSTRIAL CONTROL COMMUNICATIONS, INC.'S 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.

This datasheet may not cover all of the variations of interface applications, nor may it provide information on every possible contingency concerning installation, programming, operation, or maintenance.

The contents of this datasheet shall not become a part of or modify any prior agreement, commitment, or relationship between the customer and Industrial Control Communications, Inc. The sales contract contains the entire obligation of Industrial Control Communications, Inc. The warranty contained in the contract between the parties is the sole warranty of Industrial Control Communications, Inc., and any statements contained herein do not create new warranties or modify the existing warranty.

Any electrical or mechanical modifications to this equipment without prior written consent of Industrial Control Communications, Inc. will void all warranties and may void any UL/cUL listing or other safety certifications. Unauthorized modifications may also result in equipment damage or personal injury.

TABLE OF CONTENTS

1 Feature Summary

- 0.85” x 0.85” dimensions
- On-board full-speed USB 2.0 support
- 9X shared GPIO channels
- 3X shared pulse output channels (filter to produce analog outputs)
- 4X shared 10-bit analog input channels
- Dual serial ports with RS-485 line enable signals
- Host SPI interface
- Supports a variety of standard industrial and building-automation protocols, such as BACnet MS/TP (client & server), JCI Metasys N2 (master & slave), Modbus RTU (master, slave & sniffer), Siemens FLN etc.
- Supports configurable UART baud rates up to 115.2kbaud
- Customizable Windows®-based Configuration Studio with easy OEM branding

2 Audiences

There are two primary intended audiences for the PicoPort:

- Equipment manufacturers who already have (or can provide) an “intelligent” device that supports an existing UART. Any standard protocol (such as Modbus RTU master or slave etc.) or even custom protocol can be configured on the PicoPort in order to read/write data from the host CPU and make it accessible to the outside world.
- Sensor manufacturers that currently provide traditional “non-communicating” sensors. As the PicoPort supports a variety of on-board analog and digital I/O, it is capable of being configured to provide access to these physical I/O channels via any supported protocol. Firmware development is currently under way to incorporate advanced features related to sensor applications, such as selectable linearization for thermistors and RTDs, etc.

3 Customer Mounting

The PicoPort is directly mounted to the OEM customer’s circuit board, and the customer chooses their own physical layer interface(s). The benefits of allowing the customer to incorporate their own physical layer(s) include:

- The customer can choose their own physical layer type (4-wire or 2-wire RS-485, RS-232, SPI, TTL, isolated or non-isolated, etc.)
- The customer can choose their external interface/terminal blocks for consistent appearance to end-users.

- There are no issues with an included physical layer not being coplanar with the remainder of the customer's carrier board (which would otherwise result in having one terminal block vertically offset from all the others by ¼", etc.)
- The PicoPort can be internally mounted on the customer's board at any convenient location: it need not be located adjacent to an enclosure exit point for connector access.

For electrical and mechanical stability, it is recommended that the PicoPort be soldered directly to the host circuit board, consistent with any typical through-hole semiconductor component.

Note that when designing a 4-wire or 2-wire RS-485 physical layer, it is strongly recommended to include an additional customer connection for signal common that is connected to the ground terminal of the RS-485 transceiver. To further improve the robustness of communications, we recommend that the RS-485 transceiver circuitry be galvanically isolated from other circuitry.

4 Development Kit

A development kit is available which facilitates the ability to quickly integrate with the target OEM equipment (refer to Figure 1). The development board supports a variety of selectable communication interfaces, as well as a breadboard area for convenient hardware prototyping. Refer to the separate *PicoPort Development Kit User's Manual* for more information.

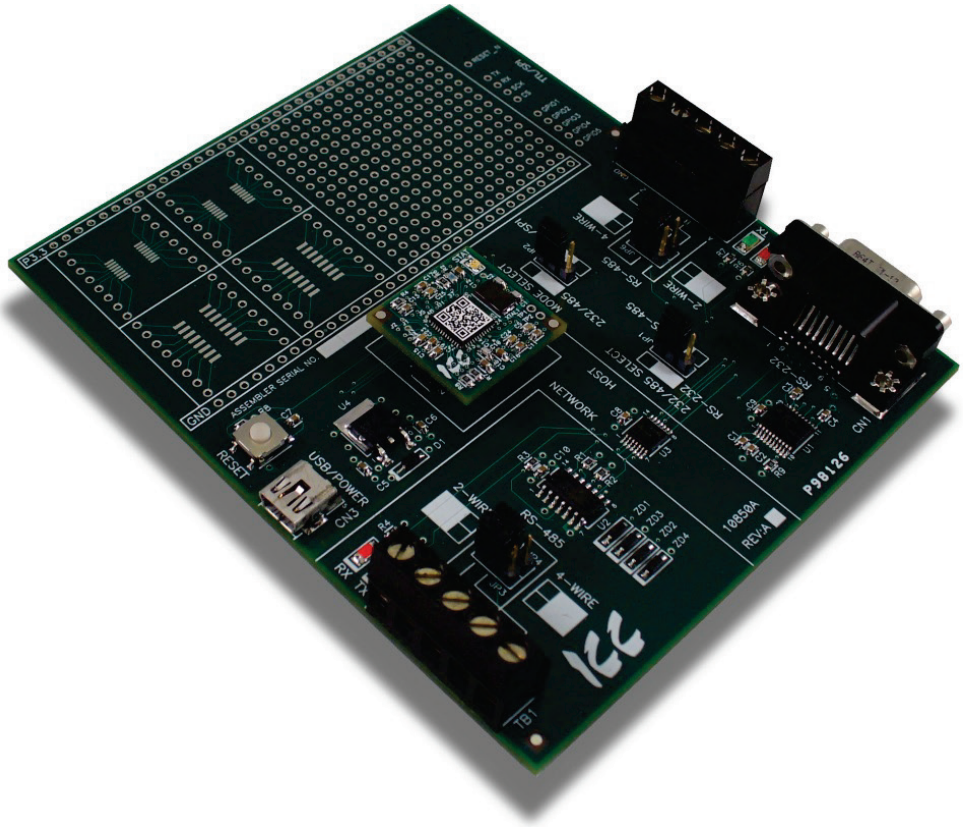


Figure 1: PicoPort Development Kit

5 Gateway Concepts

The PicoPort can be thought of as a gateway-on-a-chip. It provides simultaneous support for many different communication protocols, allowing complex interchanges of data on otherwise unsupported networks. The PicoPort is configured using the ICC Configuration Studio.

The heart of the PicoPort's gateway operation is its internal database. The database is an 8 KB, byte-wise addressable data array. At the end of the 8KB address space, the database also includes 32 bytes of Network Configuration Parameters, 32 bytes of Protocol-Specific Configuration Parameters, and 64 bytes of Persistent User Parameters. This gives a total size of 8320 bytes for the entire database, referred to as DB_{Size} in the protocol driver manuals. The database allows data to be routed from any supported network, including I/O data, to any other supported network. Data may be stored into the database in either big-endian style (meaning that if a 16-bit or 32-bit value is stored in the database, the most significant byte will start at the lowest address) or little-endian style (meaning that if a 16-bit or 32-bit value is stored in the database, the least significant byte will start at the lowest address).

The other fundamental aspect of the PicoPort is the concept of a configurable "service object". A service object is used for any master/client protocol to describe what service (read or write) is to be requested on the network. The PicoPort will cycle through the defined service objects in a round-robin fashion; however, the device does implement a "write first" approach. This means that the PicoPort will perform any outstanding write services before resuming its round-robin, read request cycle.

Additionally, the database and service objects provide the added benefit of "data mirroring", whereby current copies of data values (populated by a service object) are maintained locally within the device itself. This greatly reduces the request-to-response latency times on the various networks, as requests (read or write) can be entirely serviced locally, thereby eliminating the time required to execute a secondary transaction on a different network.

In order to facilitate the free scaling and conversion of native data values, a user-configurable "multiplier" and "data type" exist for some network configurations. All network values are scaled by a multiplier prior to being stored into the database or after being retrieved from the database. The data type is used to determine how many bytes are allocated for the value in the database, whether the value should be treated as signed or unsigned, and whether the value should be interpreted as an integer or a floating point number upon retrieval from the database.

A typical use of the multiplier feature is to preserve the fractional components of a network value for insertion into the database. For example, if the floating-point value "3.19" is read by the device from a remote BACnet device, then we could use a multiplier value of 0.01 to preserve all of the significant digits of this value: the network representation (3.19) will be divided by the multiplier value (0.01) to obtain a resultant value of 319, which will then be inserted into the database. Similarly, when a value in the database corresponding to a specific service object is changed (which therefore requires that this updated value be written to the associated remote device on the network), the service object's multiplier value will first be multiplied by the database value in order to obtain the resultant network value. For example, if 3000 is written to the database at a location corresponding to a certain service object on the other port, and that service object's multiplier value is 0.1, then the database value (3000) will be multiplied by the multiplier value (0.1) to obtain the resultant network value of 300.0, which will then be written to the network as a native floating point value.

An appropriate data type should be selected based on the range of the network data values. For example, if the value of an Analog Output on a remote BACnet device can vary from –500 to 500, a 16-bit signed data type should be used. If the value can only vary from 0 to 150, for example, an 8-bit unsigned data type may be used. Care must be taken so that a signed data type is selected if network data values can be negative. For example, if 0xFF is written to the database at a location corresponding to a service object with an 8-bit unsigned data type, the resultant network value will be 255_{10} (assuming a multiplier of 1). However, if 0xFF is written to the database at a location corresponding to a service object with an 8-bit signed data type, the resultant network value will be -1_{10} (again, assuming a multiplier of 1). It is also important to select a data type large enough to represent the network data values. For example, if a value of 257 is read by the device from a remote device and the data type corresponding to that service object is 8-bit unsigned, the value that actually will be stored is 1 (assuming a multiplier of 1). This is because the maximum value that can be stored in 8-bits is 255. Any value higher than this therefore results in overflow.

The PicoPort also provides a powerful data-monitoring feature that allows the user to view and edit its database in real time, as well as view the status of service objects via the *ICC Configuration Studio*'s Database panel when connected via USB to a PC.

6 Configuration

6.1 Overview

Module configuration is performed via USB with the Windows®-based *ICC Configuration Studio*. The USB interface does not need to be accessible to the end-user if the OEM provides all configuration of the module at manufacture time. The *ICC Configuration Studio* allows OEM customers and/or end-users to define and map their host information and physical I/O to the network(s) of their choice. The *ICC Configuration Studio* is also easily customizable for OEMs, allowing them to insert their own logos and product names etc. without any interaction from ICC. In this way, if the OEM wishes to allow their end-users to configure their device's network characteristics via USB, then the end-user perceives the device as being cohesive with the manufacturer's overall product.

If, on the other hand, the OEM wishes to provide a fixed object definition for the various supported networks, then they can load this configuration into the module themselves at manufacture time: various run-time configuration parameters exist that allow the selection of the external network characteristics via an intelligent on-board interface (such as the host device's keypad etc.). In this scenario, there would be no reason for an end-user to be exposed to the *ICC Configuration Studio*, as the OEM has already done all of the work for them.

6.2 ICC Configuration Studio

This section will provide only a brief introduction to the configuration concepts of the *ICC Configuration Studio*. For more detailed information on how to install and use the Configuration Studio, refer to the separately-available training resources.

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 it in the **Available Devices** panel.
- Right-clicking on it 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 it 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.

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.

Importing a Configuration from a Project File

An existing project file can be imported into the currently-active project. Click **File…Import Project,** and then select the desired *.icsproj file. The contents of the imported file will be merged with the active project.

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.

Updating Firmware

The studio automatically manages firmware updates when going online with a device and downloading a configuration to a device. Do not power off the device 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**.

Interacting with the Database

To interact with a device's database, select the device in the **Project** panel and then select the **Database** panel. If the **Database** panel is not visible, it can be enabled via **View…Database**. When an online device is selected, data values are updated from the device in real-time, and values can be edited by double-clicking the desired location in the database.

Diagnostics

To monitor the status of service objects, select the device in the **Project** panel and then select the **Diagnostics** panel. If the **Diagnostics** panel is not visible, it can be enabled via **View…Diagnostics**. When an online device is selected, diagnostics information is updated from the device in real-time. Individual diagnostics entries can be selected by clicking on them in the list, and multiple entries can be selected by either <CTRL>+clicking on them (to select them individually) or <SHIFT>+clicking on them (to select a range of entries). Counter values of all currently-selected diagnostics entries can be reset by clicking the **Reset Selected Counters** button.

General Configuration Process

To configure a device, add the desired protocols for the various ports, configure the communication settings (baud rate, parity, address, timeout, and scan rate/response delay etc.), and configure any objects associated with the respective protocols. 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.

6.2.1 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 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.

6.2.2 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.

Product ID

Defines a 16-bit, hexadecimal product ID for the device. This sets the value of the Product ID network configuration parameter and can be used to uniquely identify different OEM products or configurations.

Database Endianness Selection

Select the desired endianness for how data will be stored in the device's internal database for multi-byte data types. For more information on database endianness, refer to Appendix A: Database Endianness.

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 (due to dynamic configuration during startup), set this to **Automatic**.

Auto Run

Check this to allow the device to enter run mode automatically. If this is unchecked, the device will stay in configuration mode until the Run Mode configuration parameter is set to run. Note that the network driver is not started until the device enters run mode.

6.2.2.1 Configuration Locking Settings

The configuration locking settings allows a user to lock the device's configuration for reading and writing. When the device's configuration is locked, a user must enter the correct credentials in the ICC Configuration Studio to view or modify the configuration on the device.

Enable Lock

Check this to enable configuration locking on the device. After a configuration which enables this setting has been downloaded to the device, the device's configuration will be locked.

User Name

Enter the user name required to unlock the configuration on the device for reading and writing.

Password

Enter the password required to unlock the configuration on the device for reading and writing.

6.2.2.2 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. 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.

Database Value

This option configures the status LED to be fully controlled by a value located in the device's internal 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 1 lists the supported LED states.

Table 1: Status LED States

Value	LED State
0	Off
1	Green On
2	Red On
3	Green Flashing
4	Red Flashing
5…255	Off

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.

Database Address

Defines the address in the device's database used to control the status LED.

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

6.2.3 Host Settings

A Host port can be added to the device configuration which allows communication to a host processor. A variety of protocol drivers are available for the host port, depending on the host's communication capabilities (refer to section 7 for further details). Only one protocol can be selected for the host port.

6.2.4 Network Settings

The Network port is intended for communication to the "outside world". While the protocol drivers that can be assigned to the network port are for the most part the same as the host port (refer to section 7), the main difference is that multiple drivers can be included. The default protocol is designated via the Default Network Protocol device setting.

6.2.5 USB Virtual COM Port Settings

The device can be configured to enumerate as a USB virtual COM port, providing direct serial communications between the device 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 device 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 device is in this mode all other functionality of the device 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 device will communicate with the PC over the virtual COM port using the settings configured on the associated serial port. This allows the device to communicate with the PC using any of the supported serial port protocols. Note that the physical serial port is disabled when the device 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 device 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.

6.2.6 USB Serial Capture Window

The USB Serial Capture Window allows you to connect to a device's USB Virtual COM port to view and save network packets captured by the device. The device'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 device'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 device 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 device must first be configured with the appropriate USB Virtual COM port settings as described above. Once configured, the device 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.

6.2.7 Batch Update Mode

The ICC 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 device, update the firmware, update the configuration if a matching configuration is found in the project, and then go offline with the device. 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 device.

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.icsproj” 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 device 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.

6.2.8 I/O Settings

6.2.8.1 Overview

GPIO pins are available for implementations in which the module is expected to interact with physical process signals on the host device. While a variety of different capabilities (analog input, digital output, pulse counter, etc.) exist, note that not all capabilities are available on all GPIO pins. Refer to section 8.6 for a summary of the hardware capabilities of each GPIO pin.

6.2.8.1.1 Analog Input

The analog input I/O type samples analog voltage levels between 0 and VCC on the input pin. A 10-bit ADC is used to encode the voltage levels to a numeric value between 0 and 1023. The raw ADC value may be scaled before being stored into the database using a “y = mx + b” -style function, where y is the resultant database value, x is the raw ADC value, m is the multiplier and b is the offset.

Database Address

Defines the database location where the (post-modified) sampled analog input value resides. The configuration studio will not allow entry of a database address that will cause the value to run past the end of the database. The highest valid database address, therefore, depends on the designated “Data Type”.

Data Type

Specifies how the value will be stored in the database. This defines how many bytes will be allocated, whether the value should be treated as signed or unsigned, and whether the value should be interpreted as an integer or a floating point number. Select the desired data type from this dropdown menu.

Multiplier

The amount that the raw sampled analog input value is scaled by prior to being stored into the database (i.e. “m” in the “y = mx + b” equation).

Offset

The amount that is added to the scaled analog input value prior to being stored into the database (i.e. “b” in the “y = mx + b” equation).

6.2.8.1.2 Pulse (Analog) Output

The pulse output I/O type can generate a 0.06 Hz - 6 MHz pulse waveform on the output pin with a duty cycle between 0.00% and 100.00%. Two modulation modes are available: pulse width modulation and frequency modulation.

In pulse width modulation mode, a duty cycle between 0.00% and 100.00% corresponds to a raw value between 0 and 10,000. The frequency is fixed in this mode to a user-definable value between 0.06 Hz and 6 MHz. However, note that frequencies above 600 KHz lose duty cycle precision as the frequency increases. For example, at 6 MHz, there are only 10 distinct duty cycle states.

In frequency modulation mode, a frequency between 0.06 Hz and 6 MHz corresponds to a raw value between 6 and 600,000,000. A value of 0 may be written to disable the output. In this mode, the duty cycle is fixed at 50%.

The database value may be scaled using an “x = (y - b) / m” inverse slope/intercept -style function to produce the raw value used for the duty cycle or frequency, where y is the database value, x is the raw value, m is the multiplier and b is the offset.

Database Address

Defines the database location where the (pre-modified) duty cycle or frequency value resides. The configuration studio will not allow entry of a database address that will cause the value to run past the end of the database. The highest valid database address, therefore, depends on the designated “Data Type”.

Data Type

Specifies how the value will be stored in the database. This defines how many bytes will be allocated, whether the value should be treated as signed or unsigned, and whether the value should be interpreted as an integer or a floating point number. Select the desired data type from this dropdown menu.

Multiplier

The amount that the database value and offset difference is divided by to produce the raw duty cycle or frequency value (i.e. “m” in the “x = (y - b) / m” equation).

Offset

The amount that is subtracted from the database value prior to being scaled by the multiplier to produce the raw duty cycle or frequency value (i.e. “b” in the “x = (y - b) / m” equation).

Modulation Mode

Selects whether the pulse width or frequency of the pulse waveform will be modulated in relation to the database value. When frequency modulation is selected, the duty cycle is fixed at 50%.

PWM Frequency

Specifies the frequency of the PWM waveform (0.06 Hz - 6 MHz). This option is only available when pulse width modulation is selected as the “Modulation Mode”.

6.2.8.1.3 Digital Input

The digital input I/O type samples a high or low voltage level on the input pin. A sampled high level produces a value of “1” for each bit selected in the bitmask when the polarity is set to “Normal”, or a value of “0” when the polarity is set to “Reverse”. A sampled low level produces a value of “0” for each bit selected in the bitmask when the polarity is set to “Normal” or a value of “1” when the polarity is set to “Reverse”.

Database Address

Defines the database address where the digital input’s value bit(s) reside.

Data Type

Fixed at “8-Bit Unsigned”

Bitmask

Specifies which bit(s) in the byte designated by the “Database Address” that the digital input maps to. It is possible to map a single digital input to multiple bits within the designated database location. All bits designated by the bitmask will correspond to the most-recently sampled value of the GPIO pin: they will either match, or be the inversion of the GPIO’s physical state, depending on the “Polarity” setting.

Polarity

Indicates the relationship between the physical state of the GPIO pin and the logical value(s) stored in the bits designated by the “Bitmask”. If the designated polarity setting is “Normal”, then the database bits will match the physical state of the GPIO pin (“1” when the pin is sampled as “high”, and “0” when the pin is sampled as “low”.) This relationship is reversed when the designated polarity setting is “Reverse”.

6.2.8.1.4 Digital Output

The digital output I/O type generates a high or low voltage level on the output pin. When all bits selected in the bitmask are “1”, a high level is produced when the polarity is set to “Normal”, and a low level is produced when the polarity is set to “Reverse”. Otherwise (when one or more bits are “0”), a low level is produced when the polarity is set to “Normal”, and a high level is produced when the polarity is set to “Reverse”.

Database Address

Defines the database address where the digital output’s value bit(s) reside.

Data Type

Fixed at “8-Bit Unsigned”

Bitmask

Specifies which bit(s) in the byte designated by the “Database Address” that map to the digital output. It is possible to map multiple bits within the designated database location to a single digital output. All bits designated by the bitmask will dictate the driven value of the GPIO pin: if all bits are “1” then the output value will be high, otherwise the output value will be low (assuming “Normal” polarity.)

Polarity

Indicates the relationship between the physical state of the GPIO pin and the logical value(s) stored in the bits designated by the “Bitmask”. If the designated polarity setting is “Normal”, then the GPIO pin will be high when all bits in the bitmask are “1”. This relationship is reversed when the designated polarity setting is “Reverse”.

6.2.8.1.5 Pulse Counter

The pulse counter I/O type is an interrupt-driven input which can be configured in combination with any other I/O type. The interrupt-driven pulse count is then sampled at the same rate as other I/O inputs and accumulated in a database location. The count can be triggered to increment on rising edge transitions, falling edge transitions, or both depending on the selected mode. The pulse counter also features a debounce time to ignore pulses shorter than a specified duration.

Note that because the pulse counter is interrupt-driven, the device could become unresponsive during continuous, high-frequency pulses. Care should be taken to avoid applying a continuous, high-frequency signal to an I/O pin configured as a pulse counter.

Database Address

Defines the database location where the accumulated pulse count value resides. The configuration studio will not allow entry of a database address that will cause the value to run past the end of the database. The highest valid database address, therefore, depends on the designated "Data Type".

Data Type

Specifies how the accumulated data will be stored in the database. This defines how many bytes will be allocated for the value. Select the desired data type from this dropdown menu.

Mode

The mode selects the event which causes the pulse counter to increment. Select between "Pin Change" (both rising & falling edge), "Rising Edge", or "Falling Edge".

Debounce Time

Specifies the minimum amount of time (in milliseconds) a pulse must remain at either a high or low level in order to cause the pulse counter to increment.

6.2.9 Internal Logic Settings

6.2.9.1 Initial Persistent Values

6.2.9.1.1 Overview

The PicoPort 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 PicoPort's persistent user parameters, refer to Section 6.4.

6.2.9.1.2 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 8-bit, 16-bit, or 32-bit value is written to the corresponding persistent database address(es). 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:

Database Address

Enter the starting database address in the persistent memory block where the first data element of this initializer object will begin. The maximum allowable database address depends on the designated Data Type.

Data Type

Specifies how the initializer value will be stored in the database. This defines how many bytes will be allocated, whether the value should be treated as signed or unsigned, and whether the value should be interpreted as an integer or a floating point number. Select the desired data type from this dropdown menu.

Value

Enter the value that each database address encompassed by this initializer object will be written to when the persistent memory is initialized.

Length

Enter the number of data elements for this initializer object. The total number of database bytes modified by an initializer object is determined by the Length multiplied by the number of bytes in the selected Data Type (1, 2 or 4 for 8-bit, 16-bit and 32-bit, respectively).

6.2.9.2 Fail-safe Values

6.2.9.2.1 Overview

The device can be configured to perform a specific set of actions when network communications are lost. This allows each address in the database 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 specific protocol's driver manual for further information.

Note that this timeout feature is only used with slave/server protocols: this is not the same as the Timeout time used for service objects in master/client protocols.

There are two separate elements that comprise the timeout configuration:

- The timeout time
- Timeout Object configuration

6.2.9.2.2 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 device-wide 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.

For some protocols, the timeout time is set by the master device (PLC, scanner, etc.), and a timeout time setting is therefore not provided in the Configuration Studio's driver configuration. Additionally, not all protocols support timeout detection: refer to the specific protocol's driver manual for more information.

6.2.9.2.3 Timeout Object Configuration

A timeout object is used by the device as part of the timeout processing to set certain addresses of the database to "fail-safe" values. When a timeout event is triggered by a protocol, the timeout objects are parsed and the designated 8-bit, 16-bit, or 32-bit value is written to the corresponding database address(es). To add a timeout object to a device, 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:

Database Address

Enter the starting address in the database where the first data element of this timeout object will begin. The maximum allowable database address depends on the designated Data Type.

Data Type

Specifies how the timeout value will be stored in the database. This defines how many bytes will be allocated, whether the value should be treated as signed or unsigned, and whether the value should be interpreted as an integer or a floating point number. Select the desired data type from this dropdown menu.

Value

Enter the "fail-safe" timeout value that each database address encompassed by this timeout object will be automatically written with upon processing a timeout event triggered by a protocol.

Length

Enter the number of data elements for this timeout object. The total number of database bytes modified by a timeout object is determined by the Length multiplied by the number of bytes in the selected Data Type (1, 2 or 4 for 8-bit, 16-bit and 32-bit, respectively).

6.2.9.3 Database Logic

6.2.9.3.1 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 6.2.9.3.3 for more information on Enable Trigger behavior.)
- The outputs of all operations must be stored in the internal database.
- The number of bytes taken from the database (for non-constant inputs) is determined by the corresponding "Data Type" selection, starting at the designated "Database Address".
- The number of bytes written to the database (for outputs) is determined by the corresponding "Data Type" selection, starting at the designated "Database Address".

Logical Operations

- *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 *Mutiplexer* 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] × [Input 2]*.
- The *Divide* operation calculates the expression *[Input 1] / [Input 2]*.
- The *Modulo* operation calculates the expression *[Input 1] mod [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 $sin^{-1}(Input\ 1)$, where the output is in radians.
- The *Arc Cosine* operation calculates the expression $cos^{-1}(Input\ 1)$, where the output is in radians.
- The *Arc Tangent* operation calculates the expression $tan^{-1}(Input\ 1)$, 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”.

6.2.9.3.2 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 6.2.9.3.3 for more information on Enable Trigger behavior.

6.2.9.3.3 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 Value Database Address

Enter the database address which specifies the byte-size trigger value.

6.2.9.3.3.1 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 setting 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.

6.2.10 Service Objects and Diagnostics Objects

A service object is used by the device to make requests on a network when a master/client protocol is enabled. Each service object defines the services (read and/or write) that should be performed on a range of network objects of a common type. The data from read requests is mirrored in the database starting at a user-defined address (if a read function is enabled). When a value within that address range in the database changes, a write request is generated on the network (if a write function is enabled). Specific service object configuration depends on the protocol selected: refer to the specific protocol’s driver manual for further details.

Master/client drivers commonly also provide the ability to debug configured service objects while the driver is running by way of optional diagnostics objects. Where supported, diagnostics objects can be added to each service object, and a database address can be designated at which to store the status information. The diagnostics object is a 16-byte structure containing elements such as a transmission counter, receive counter, receive error counter, current status, and the last error of the defined service object. This information is detailed in Appendix B:

Diagnostics Objects. Because the diagnostics object resides in the database alongside the service object's process data, it can also be accessed over any supported network by mapping appropriate network elements to the corresponding database addresses.

Alternatively, the diagnostics objects can be viewed within the Configuration Studio by selecting a device in the **Project** panel and then clicking on the **Diagnostics** tab. Diagnostics objects are automatically added to the **Diagnostics** panel, and are disseminated and displayed in plain text for easy interpretation. For online devices, diagnostics objects are updated in real-time and all counters can be reset by selecting one or more entries in the list and clicking the **Reset Selected Counters** button.

6.3 Network Configuration Parameters

The PicoPort has a bank of internal parameters which allow dynamic configuration of the network port properties. These configuration parameters can be modified by both the host processor and the network itself, and it is possible to modify them during startup as well as while the module is running. This flexibility allows the host processor or network client to perform tasks such as changing the protocol, address, baud rate, and parity of the network port. The configuration parameters are internally mapped to the module's database and are summarized in Table 2.

Note

- All configuration parameters are 16-bit unsigned values (consuming 2 bytes in the internal database) unless otherwise noted
- Configuration parameter range checking is not performed when the parameter is modified by the host processor or from the network: range checking is performed by each specific driver during initialization, and invalid settings will result in the module transitioning to the error state with an indication of "invalid or corrupt configuration"

Table 2: PicoPort Configuration Parameters

Database Address	Parameter	Notes	
8192	Product ID	OEM-configurable Product ID (*read-only*) Default = 0x2101	
8194	Firmware Version	Value = Firmware version * 1000 (*read-only*) (for example, 2300 = V2.300)	
8196	Status Code	*Read-only* 0 = Normal 6 = USB to Serial Pass-Through Mode 7 = Invalid configuration parameters All other values = For internal use only (contact ICC)	
8198	Run Mode	0 = Startup (*read-only value*) 1 = Configuration mode (*read-only value*) 2 = Running (*read/write value*) 3 = Error (*read-only value*) 65535 (0xFFFF) = Reset (*write-only value*)	
8200	Protocol	0 = Disabled 1 = Modbus RTU master 2 = Modbus RTU slave 3 = BACnet MS/TP server 4 = BACnet MS/TP client 5…11 = Reserved 12 = Metasys N2 slave 13 = Toshiba ASD master 14 = Sullair master 15 = Modbus RTU sniffer 16 = MSA Chillgard monitor 17 = Metasys N2 master 18 = Siemens FLN slave 19 = TCS Basys master	20 = DMX-512 master 21 = DMX-512 slave 22 = M-Bus master 23 = AO Smith AIN slave 24 = AO Smith PDNP master 25 = Kele PowerTrak 26 = Kele EnGenius 27 = Siemens FLN master 28 = Toshiba Computer Link master 29 = Generic Serial Master 30 = Generic Serial Slave 253 = Host - Network Pass-Through
8202	Address	Protocol-specific	
8204	Baud Rate	Value = Baud rate / 100 (for example, 96 = 9600 baud)	
8206	Parity	0 = No parity, 1 stop bit 1 = Odd parity, 1 stop bit	2 = Even parity, 1 stop bit 3 = No parity, 2 stop bits
8208	Timeout	Value in ms	
8210	Scan Rate / Response Delay	Value in ms	
8212	Number of Retries	Protocol-specific	
8214…8223	Reserved	Reserved for future use	

Database Address	Parameter	Notes
8224…8255	Protocol-Specific Parameters	Defined by the currently-selected protocol

As shown in Table 2, shared network configuration parameters numbered 8202 and higher may have different valid ranges (or may be ignored altogether) depending on the selected protocol. There is also a range of parameters whose use is protocol-specific: their meaning and adjustment ranges are unique only to the currently-selected driver. Refer to the following tables for the settings available for each driver.

Table 3: Modbus RTU Master Parameters

Database Address	Parameter	Notes
8202	Address	Ignored
8204	Baud Rate	24…1152 (2400…115200 baud)
8206	Parity	0…3
8208	Timeout	0…65535 (0…65.535s)
8210	Scan Rate	0…65535 (0…65.535s)
8212	Number of Retries	Ignored

Table 4: Modbus RTU Slave Parameters

Database Address	Parameter	Notes
8202	Address	1…247
8204	Baud Rate	24…1152 (2400…115200 baud)
8206	Parity	0…3
8208	Timeout	0…65535 (0…65.535s)
8210	Response Delay	0…65535 (0…65.535s)
8212	Number of Retries	Ignored
8224	Word Order Override Enable	Enables the word order override for all register mapping objects 0 = Disabled 1 = Enabled
8226	Word Order	0 = Little endian word order 1 = Big endian word order

Table 5: BACnet MS/TP Server Parameters

Database Address	Parameter	Notes
8202	Address	0…127
8204	Baud Rate	96…1152 (9600…115200 baud)
8206	Parity	Ignored (fixed at no parity / 1 stop bit)
8208	APDU Timeout	0…65535 (0…65.535s)
8210	Scan Rate / Response Delay	Ignored
8212	Number of APDU Retries	0…10
8224…8239	Device Name	16-character device name as per the BACnet specification
8240	Device Instance	0…4194302 (32-bit unsigned value)
8244	Max Master	Address…127
8246	UTC Offset	-840…720 (in minutes) Ignored if real-time clock functionality is disabled
8248	Daylight Saving	0 = Off 1 = On Ignored if real-time clock functionality is disabled

Table 6: BACnet MS/TP Client Parameters

Database Address	Parameter	Notes
8202	Address	0…127
8204	Baud Rate	96…1152 (9600…115200 baud)
8206	Parity	Ignored (fixed at no parity / 1 stop bit)
8208	APDU Timeout	0…65535 (0…65.535s)
8210	Scan Rate	0…65535 (0…65.535s)
8212	Number of APDU Retries	0…10
8224…8239	Device Name	16-character device name as per the BACnet specification
8240	Device Instance	0…4194302 (32-bit unsigned value)
8244	Max Master	Address…127
8246	UTC Offset	-840…720 (in minutes) Ignored if real-time clock functionality is disabled
8248	Daylight Saving	0 = Off 1 = On Ignored if real-time clock functionality is disabled

Table 7: Metasys N2 Slave Parameters

Database Address	Parameter	Notes
8202	Address	1…255
8204	Baud Rate	Ignored (fixed at 9600 baud)
8206	Parity	Ignored (fixed at no parity / 1 stop bit)
8208	Timeout	0…65535 (0…65.535s)
8210	Response Delay	0…65535 (0…65.535s)
8212	Number of Retries	Ignored

Table 8: Toshiba ASD Master Parameters

Database Address	Parameter	Notes
8202	Address	Ignored
8204	Baud Rate	24…1152 (2400…115200 baud)
8206	Parity	0…3
8208	Timeout	0…65535 (0…65.535s)
8210	Scan Rate	0…65535 (0…65.535s)
8212	Number of Retries	Ignored

Table 9: Sullair Master Parameters

Database Address	Parameter	Notes
8202	Address	Ignored (fixed at 0)
8204	Baud Rate	Ignored (fixed at 9600 baud)
8206	Parity	Ignored (fixed at no parity / 1 stop bit)
8208	Timeout	Ignored (fixed at 500ms)
8210	Scan Rate	Ignored (fixed at 0)
8212	Number of Retries	Ignored

Table 10: Modbus RTU Sniffer Parameters

Database Address	Parameter	Notes
8202	Address	Ignored
8204	Baud Rate	24…1152 (2400…115200 baud)
8206	Parity	0…3
8208	Timeout	Ignored
8210	Scan Rate / Response Delay	Ignored
8212	Number of Retries	Ignored

Table 11: MSA Chillgard Monitor Parameters

Database Address	Parameter	Notes
8202	Address	Ignored
8204	Baud Rate	Ignored (fixed at 19200 baud)
8206	Parity	Ignored (fixed at no parity / 1 stop bit)
8208	Timeout	Ignored (fixed at 4.5s)
8210	Scan Rate / Response Delay	Ignored
8212	Number of Retries	Ignored

Table 12: Metasys N2 Master Parameters

Database Address	Parameter	Notes
8202	Address	Ignored
8204	Baud Rate	Ignored (fixed at 9600 baud)
8206	Parity	Ignored (fixed at no parity / 1 stop bit)
8208	Timeout	0…65535 (0…65.535s)
8210	Scan Rate	0…65535 (0…65.535s)
8212	Number of Retries	Ignored

Table 13: Siemens FLN Slave Parameters

Database Address	Parameter	Notes
8202	Address	1…98
8204	Baud Rate	24…1152 (2400…115200 baud)
8206	Parity	Ignored (fixed at no parity / 1 stop bit)
8208	Timeout	0…65535 (0…65.535s)
8210	Response Delay	0…65535 (0…65.535s)
8212	Number of Retries	Ignored

Table 14: TCS Basys Master Parameters

Database Address	Parameter	Notes
8202	Address	Ignored
8204	Baud Rate	24…1152 (2400…115200 baud)
8206	Parity	Ignored (fixed at no parity / 1 stop bit)
8208	Timeout	0…65535 (0…65.535s)
8210	Scan Rate	0…65535 (0…65.535s)
8212	Number of Retries	Ignored

Table 15: DMX-512 Master Parameters

Database Address	Parameter	Notes
8202	Address	Ignored
8204	Baud Rate	Ignored (fixed at 250kbaud)
8206	Parity	Ignored (fixed at no parity / 2 stop bits)
8208	Timeout	Ignored
8210	Scan Rate	0…65535 (0…65.535s)
8212	Number of Retries	Ignored

Table 16: DMX-512 Slave Parameters

Database Address	Parameter	Notes
8202	Address	1…512
8204	Baud Rate	Ignored (fixed at 250kbaud)
8206	Parity	Ignored (fixed at no parity / 2 stop bits)
8208	Timeout	0…65535 (0…65.535s)
8210	Response Delay	Ignored
8212	Number of Retries	Ignored

Table 17: M-Bus Master Parameters

Database Address	Parameter	Notes
8202	Address	Ignored
8204	Baud Rate	3…384 (300…38400 baud)
8206	Parity	Ignored (fixed at even parity / 1 stop bit)
8208	Timeout	0…65535 (0…65.535s)
8210	Scan Rate	0…65535 (0…65.535s)
8212	Number of Retries	Ignored

Table 18: AO Smith AIN Slave Parameters

Database Address	Parameter	Notes
8202	Address	Ignored (fixed at 16)
8204	Baud Rate	192…384 (19200…38400 baud)
8206	Parity	Ignored (fixed at no parity / 1 stop bit)
8208	Timeout	Ignored
8210	Response Delay	Ignored
8212	Number of Retries	Ignored

Table 19: AO Smith PDNP Master Parameters

Database Address	Parameter	Notes
8202	Address	Ignored
8204	Baud Rate	Ignored (fixed at 19200 baud)
8206	Parity	Ignored (fixed at no parity / 1 stop bit)
8208	Timeout	0…65535 (0…65.535s)
8210	Scan Rate	0…65535 (0…65.535s)
8212	Number of Retries	Ignored

Table 20 : Siemens FLN Master Parameters

Database Address	Parameter	Notes
8202	Address	Ignored
8204	Baud Rate	24…1152 (2400…115200 baud)
8206	Parity	Ignored (fixed at no parity / 1 stop bit)
8208	Timeout	0…65535 (0…65.535s)
8210	Scan Rate	0…65535 (0…65.535s)
8212	Number of Retries	Ignored

Table 21: Toshiba Computer Link Master Parameters

Database Address	Parameter	Notes
8202	Address	Ignored
8204	Baud Rate	3…1152 (300…115200 baud)
8206	Parity	0…3
8208	Timeout	0…65535 (0…65.535s)
8210	Scan Rate	0…65535 (0…65.535s)
8212	Number of Retries	Ignored

Table 22: Generic Serial Master Parameters

Database Address	Parameter	Notes
8202	Address	Ignored
8204	Baud Rate	3…1152 (300…115200 baud)
8206	Parity	0…3
8208	Timeout	0…65535 (0…65.535s)
8210	Scan Rate	0…65535 (0…65.535s)
8212	Number of Retries	Ignored

Table 23: Generic Serial Slave Parameters

Database Address	Parameter	Notes
8202	Address	Ignored
8204	Baud Rate	3…1152 (300…115200 baud)
8206	Parity	0…3
8208	Timeout	0…65535 (0…65.535s)
8210	Response Delay	0…65535 (0…65.535s)
8212	Number of Retries	Ignored

Table 24: Host - Network Pass-Through Parameters

Database Address	Parameter	Notes
8202	Address	Ignored
8204	Minimum Baud Rate	24…1152 (2400…115200 baud)
8206	Parity	Ignored
8208	Timeout	Ignored
8210	Response Delay	Ignored
8212	Number of Retries	Ignored

6.4 Persistent User Parameters

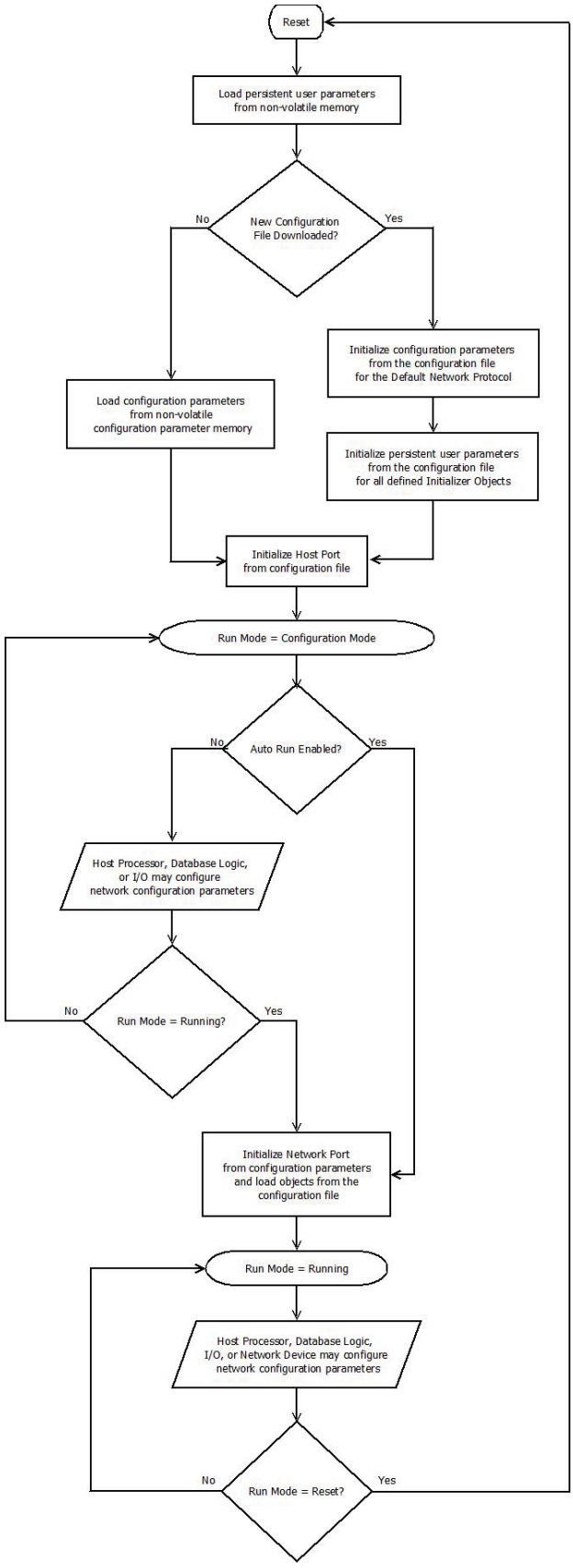
The PicoPort has a block of general-purpose, persistent memory mapped into its internal database. This is useful for data which must persist across power cycles and reboots, such as calibration data or custom serial numbers. The persistent memory block consists of 64 bytes and begins at database address 8256. Aside from their persistence, these database locations behave like any other database location. Any object may be mapped to the persistent memory database locations using any data type.

Care must be taken, however, when selecting data to be mapped into the persistent memory block. Because this memory is backed by internal flash storage, it is susceptible to write cycle limitations of flash technology. Therefore, it is important to only map data which changes infrequently to these locations.

To ease OEM programming, the PicoPort supports the definition of initial values for the persistent memory locations as part of the configuration of the device. This allows the PicoPort to be fully programmed by an OEM by simply downloading a single configuration file. For more information on initializing persistent values, refer to Section 6.2.9.1.

6.5 Initialization Overview

Because the PicoPort can function with or without a host processor, a variety of configuration options are provided. Configuration Studio settings such as Auto Run and the Default Network Protocol are used in conjunction with the configuration parameters to define the behavior of the module during initialization. This flowchart details the initialization steps that the PicoPort performs during startup.



6.6 I/O and Database Logic Scan Rate

The Configuration Studio provides a configurable “scan rate” parameter for all GPIO and database logic. Refer to sections 6.2.8 and 6.2.9.3. These settings are found in the **I/O Settings** panel when “I/O” is selected in the **Project** panel, and in the **Database Logic Settings** panel when “Database Logic” is selected in the **Project** panel. While the general behavior of these scan rate settings are similar and therefore will be jointly discussed here, note that the I/O processing and database logic processing are performed in separate threads in the PicoPort’s realtime operating system (RTOS), and therefore are unrelated to, and independent of, each other in practice.

The GPIO scan rate applies regardless of whether a GPIO pin is configured as a digital input, digital output, analog input or pulse (analog) output. In both the I/O and database logic cases, all processing is performed “in bulk”, after which a delay of “scan rate” number of ms is inserted before a subsequent processing activity is performed. This cycle (process / wait / process, etc.) is then performed forever (refer to Figure 2).

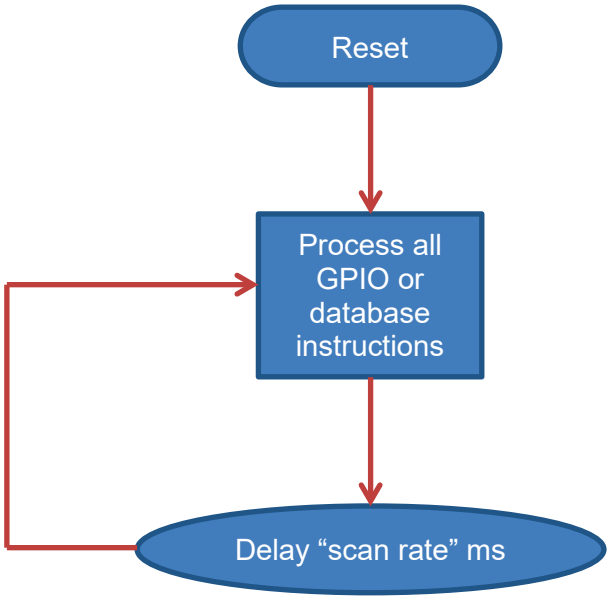


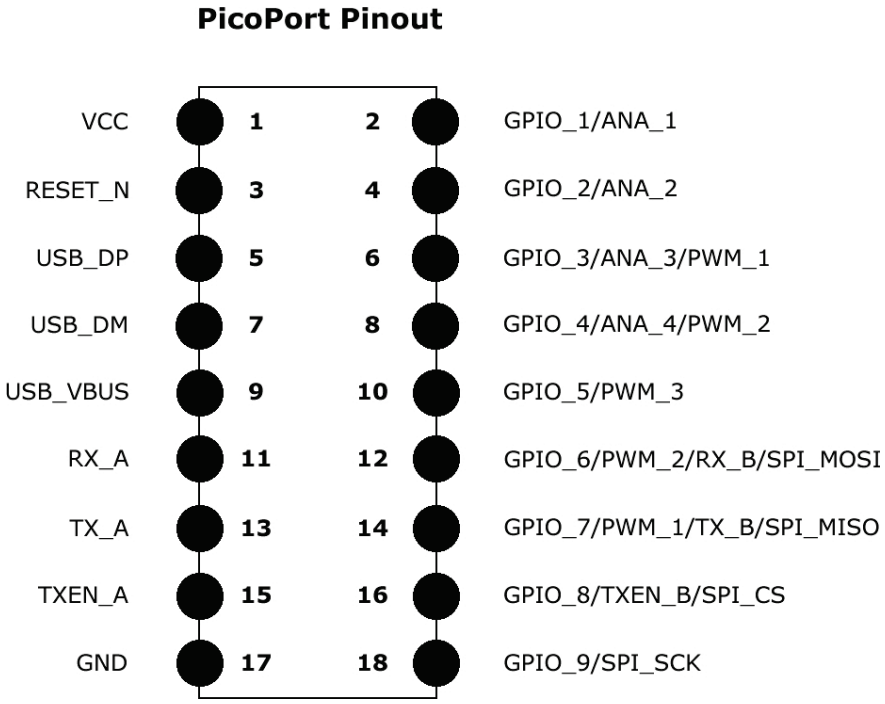
Figure 2: GPIO / Database Logic Scan Cycle

7 Serial Drivers

The PicoPort supports a variety of serial drivers on its Host and Network ports. For a list of supported protocols, refer to the *PicoPort Supported Drivers List*. For detailed information on each protocol, refer to the specific protocol's driver manual.

8 Hardware Specifications

8.1 Pinout

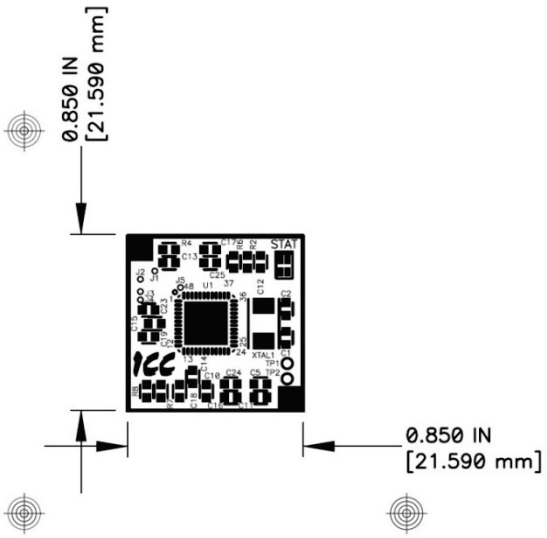


8.2 Header Interface

The module uses an 18-position (2x9) DIP header with 2.00mm spacing and 3.20mm lead length (Samtec part #TMMH-109-01-F-DV). Refer to the appropriate Samtec documentation for recommended PCB layout.

8.3 Dimensions

0.85” x 0.85” x 0.35”



8.4 Environmental Specifications

Item	Specification
Operating Environment	Indoors, less than 1000m above sea level, do not expose to corrosive / explosive gasses
Operating Temperature	-20 ~ +60°C (-4 ~ +140°F)
Storage Temperature	-40 ~ +85°C (-40 ~ +185°F)
Relative Humidity	20% ~ 90% (without condensation)
Vibration	5.9m/s^2 {0.6G} or less (10 ~ 55Hz)
Grounding	Non-isolated, referenced to power ground

This device is lead-free / RoHS-compliant.

8.5 Indicators

The module includes a single dichromatic (red/green) “status” LED. This status LED can be in one of the following three states:

Startup: The LED flashes a red/green sequence on startup.

Normal Operation: The LED is green and reflects one of the following two possible indications:

Solid green .....the module is operating normally without USB connection

Blinking green.....the module is operating normally with USB connection

Error Indication: If an error is detected, the status LED is red and blinks an error code. The number of sequential blinks (followed by 2s of OFF time) indicates the nature of the error:

Error Code	Meaning
1...5	For internal use: contact ICC for assistance
6	USB to Serial Pass-Through mode
7	Invalid or corrupt configuration
8...10	For internal use: contact ICC for assistance

8.6 Pin Descriptions

Pin #1 (VCC)

Internal voltage supervisor / power-on reset controller ensures correct module operation during power-up / power-down sequences.

Input/output..... Input
Supply voltage ..... 3.3VDC ± 5%
Typical current consumption 46mA (0.15W)

Pin #3 (RESET_N)

Logic low-level module reset signal. Internally pulled-up, therefore this pin can be left disconnected if no hardware reset control from the host system is necessary. Use open-collector type drive when interfacing to 5V control signal.

Input/output..... Input
5V tolerant ..... No
V_{IL} range ..... -0.3V to +0.8V
V_{IH} range..... 2.5V to 3.6V

Pin #5 (USB_DP)

USB Data+ signal. Connect to USB port, pin 3.

Input/output..... Bidirectional

Pin #7 (USB_DM)

USB Data- signal. Connect to USB port, pin 2.

Input/output..... Bidirectional

Pin #9 (USB_VBUS)

USB bus voltage monitor. Connect to USB port, pin 1.

Input/output..... Input

Pin #11 (RX_A)

Network-side receive data line. Connect to network transceiver's "receive data" pin.

Input/output..... Input
5V tolerant ..... Yes
V_{IL} range ..... -0.3V to +0.8V
V_{IH} range..... 2.0V to 5.5v

Pin #12 (GPIO_6/PWM_2/RX_B/SPI_MOSI)

Multi-function capable pin that can be configured with one or more of the following functions:

- General-purpose digital input

- General-purpose digital output
- Pulse output (filter to produce analog output) (PWM module shared with GPIO_4)
- Pulse counter (may be configured in combination with other I/O functions)
- Receive data

When configured as a general-purpose digital input, this pin is internally pulled-up and the following characteristics apply:

Input/output..... Input
5V tolerant ..... Yes
V_{IL} range ..... -0.3V to +0.8V
V_{IH} range..... 2.0V to 5.5V

When configured as a general-purpose digital output, the following characteristics apply:

Input/output..... Output
V_{OL} (max) 0.4V
V_{OH} (min)..... VCC-0.4V
I_{OH} (max) 4mA
I_{OL} (max) ..... -4mA

When configured as a pulse output, the following characteristics apply:

Input/output..... Output
Voltage range ..... 0V to 3.3V
Waveform frequency..... 0.06 Hz - 6 MHz
Duty cycle range…..... 0.00% to 100.00%
Waveform polarity…..... High at start of period
Waveform alignment…..... Left aligned

When configured as a pulse counter, this pin may be configured in combination with another I/O function. When configured with an additional function, the characteristics for that function apply to this pin. Otherwise, the pin is internally pulled-up and the following characteristics apply:

Input/output..... Input
5V tolerant ..... Yes
V_{IL} range ..... -0.3V to +0.8V
V_{IH} range..... 2.0V to 5.5V

When configured as receive data, the following characteristics apply:

Host-side receive data line. When using host communications, connect to host device's logic-level transmit line (when interfacing directly to host CPU) or host-side transceiver's "receive data" pin.

Input/output..... Input
5V tolerant ..... Yes
V_{IL} range ..... -0.3V to +0.8V
V_{IH} range..... 2.0V to 5.5v

Pin #13 (TX_A)

Network-side transmit data line. Connect to network transceiver's "transmit data" pin.

Input/output..... Output
V_{OL} (max)..... 0.4V
V_{OH} (min)..... VCC-0.4V
I_{OH} (max)..... 4mA
I_{OL} (max) ..... -4mA

Pin #14 (GPIO_7/PWM_1/TX_B/SPI_MISO)

Multi-function capable pin that can be configured with one or more of the following functions:

- General-purpose digital input
- General-purpose digital output
- Pulse output (filter to produce analog output) (PWM module shared with GPIO_3)
- Pulse counter (may be configured in combination with other I/O functions)
- Transmit data

When configured as a general-purpose digital input, this pin is internally pulled-up and the following characteristics apply:

Input/output..... Input
5V tolerant ..... Yes
V_{IL} range ..... -0.3V to +0.8V
V_{IH} range..... 2.0V to 5.5V

When configured as a general-purpose digital output, the following characteristics apply:

Input/output..... Output
V_{OL} (max)..... 0.4V
V_{OH} (min)..... VCC-0.4V
I_{OH} (max)..... 4mA
I_{OL} (max) ..... -4mA

When configured as a pulse output, the following characteristics apply:

Input/output..... Output
Voltage range ..... 0V to 3.3V
Waveform frequency..... 0.06 Hz - 6 MHz
Duty cycle range…..... 0.00% to 100.00%
Waveform polarity…..... High at start of period
Waveform alignment…..... Left aligned

When configured as a pulse counter, this pin may be configured in combination with another I/O function. When configured with an additional function, the characteristics for that function apply to this pin. Otherwise, the pin is internally pulled-up and the following characteristics apply:

Input/output..... Input
5V tolerant ..... Yes
V_{IL} range ..... -0.3V to +0.8V
V_{IH} range..... 2.0V to 5.5V

When configured as transmit data, the following characteristics apply:

Host-side transmit data line. When using host communications, connect to host device's logic-level receive line (when interfacing directly to host CPU) or host-side transceiver's "transmit data" pin.

Input/output ..... Output
V_{OL} (max) 0.4V
V_{OH} (min) VCC-0.4V
I_{OH} (max) 8mA
I_{OL} (max) ..... -8mA

Pin #15 (TXEN_A)

Network-side RS-485 transceiver driver enable line. Connect to RS-485 transceiver "driver enable" pin when using RS-485 based network. Signal is HI when data is being transmitted on pin TX_A, and LO at all other times.

Input/output ..... Output
V_{OL} (max) 0.4V
V_{OH} (min) VCC-0.4V
I_{OH} (max) 4mA
I_{OL} (max) ..... -4mA

Pin #16 (GPIO_8/TXEN_B/SPI_CS)

Multi-function capable pin that can be configured with one or more of the following functions:

- General-purpose digital input
- General-purpose digital output
- Pulse counter (may be configured in combination with other I/O functions)
- RS-485 transceiver enable
- SPI chip select

When configured as a general-purpose digital input, this pin is internally pulled-up and the following characteristics apply:

Input/output ..... Input
5V tolerant ..... Yes
V_{IL} range ..... -0.3V to +0.8V
V_{IH} range ..... 2.0V to 5.5V

When configured as a general-purpose digital output, the following characteristics apply:

Input/output ..... Output
V_{OL} (max) 0.4V
V_{OH} (min) VCC-0.4V
I_{OH} (max) 4mA
I_{OL} (max) ..... -4mA

When configured as a pulse counter, this pin may be configured in combination with another I/O function. When configured with an additional function, the characteristics for that function apply to this pin. Otherwise, the pin is internally pulled-up and the following characteristics apply:

Input/output ..... Input
5V tolerant ..... Yes

V_{IL} range ..... -0.3V to +0.8V
V_{IH} range ..... 2.0V to 5.5V

When configured as a RS-485 transceiver enable, the following characteristics apply:

Host-side RS-485 transceiver enable line. Connect to RS-485 transceiver "driver enable" pin when using RS-485 based host communications. Signal is HI when data is being transmitted on pin TX_B, and LO at all other times.

Input/output ..... Output
V_{OL} (max) 0.4V
V_{OH} (min) VCC-0.4V
I_{OH} (max) 4mA
I_{OL} (max) ..... -4mA

When configured as a SPI chip select, the following characteristics apply:

Host-side SPI slave chip select line. Connect to SPI master chip select pin when using SPI based host communications. The SPI master should drive this line LO when communicating to the PicoPort, and HI at all other times.

Input/output ..... Input
5V tolerant ..... Yes
V_{IL} range ..... -0.3V to +0.8V
V_{IH} range ..... 2.0V to 5.5v

Pin #18 (GPIO_9/SPI_SCK)

Multi-function capable pin that can be configured with one or more of the following functions:

- General-purpose digital input
- General-purpose digital output
- Pulse counter (may be configured in combination with other I/O functions)
- SPI serial clock

When configured as a general-purpose digital input, this pin is internally pulled-up and the following characteristics apply:

Input/output ..... Input
5V tolerant ..... Yes
V_{IL} range ..... -0.3V to +0.8V
V_{IH} range ..... 2.0V to 5.5V

When configured as a general-purpose digital output, the following characteristics apply:

Input/output ..... Output
V_{OL} (max) 0.4V
V_{OH} (min) VCC-0.4V
I_{OH} (max) 4mA
I_{OL} (max) ..... -4mA

When configured as a pulse counter, this pin may be configured in combination with another I/O function. When configured with an additional function, the characteristics for that function apply to this pin. Otherwise, the pin is internally pulled-up and the following characteristics apply:

Input/output..... Input
5V tolerant ..... Yes
V_{IL} range ..... -0.3V to +0.8V
V_{IH} range..... 2.0V to 5.5V

When configured as a SPI serial clock, the following characteristics apply:

Host-side SPI serial clock line. Connect to SPI master serial clock output pin when using SPI based host communications.

Input/output..... Input
5V tolerant ..... Yes
V_{IL} range ..... -0.3V to +0.8V
V_{IH} range..... 2.0V to 5.5v

Pin #2 (GPIO_1/ANA_1), Pin #4 (GPIO_2/ANA_2)

Multi-function capable pin that can be configured with one or more of the following functions:

- General-purpose digital input
- General-purpose digital output
- Analog input
- Pulse counter (may be configured in combination with other I/O functions)

When configured as a general-purpose digital input, this pin is internally pulled-up. Use open-collector type drive when interfacing to 5V control signal. The following characteristics apply:

Input/output..... Input
5V tolerant ..... No
V_{IL} range ..... -0.3V to +0.8V
V_{IH} range..... 2.0V to 3.6V

When configured as a general-purpose digital output, the following characteristics apply:

Input/output..... Output
V_{OL} (max) 0.4V
V_{OH} (min)..... VCC-0.4V
I_{OH} (max) 4mA
I_{OL} (max) ..... -4mA

When configured as an analog input, the following characteristics apply:

Input/output..... Input
Resolution..... 10 bits (0...1023 raw value)
V_{IN} range..... 0V to VCC
Input capacitance..... 7pF typical

When configured as a pulse counter, this pin may be configured in combination with another I/O function. When configured with an additional function, the characteristics for that function apply to this pin. Otherwise, the pin is internally pulled-up and the following characteristics apply:

Input/output..... Input
5V tolerant ..... Yes
V_{IL} range ..... -0.3V to +0.8V

V_{IH} range..... 2.0V to 5.5V

Pin #6 (GPIO_3/ANA_3/PWM_1), Pin #8 (GPIO_4/ANA_4/PWM_2)

Multi-function capable pin that can be configured with one or more of the following functions:

- General-purpose digital input
- General-purpose digital output
- Analog input
- Pulse output (filter to produce analog output) (GPIO_3 PWM module shared with GPIO_7, GPIO_4 PWM module shared with GPIO_6)
- Pulse counter (may be configured in combination with other I/O functions)

When configured as a general-purpose digital input, this pin is internally pulled-up. Use open-collector type drive when interfacing to 5V control signal. The following characteristics apply:

Input/output..... Input
5V tolerant ..... No
V_{IL} range ..... -0.3V to +0.8V
V_{IH} range..... 2.0V to 3.6V

When configured as a general-purpose digital output, the following characteristics apply:

Input/output..... Output
V_{OL} (max) 0.4V
V_{OH} (min)..... VCC-0.4V
I_{OH} (max) 4mA
I_{OL} (max) ..... -4mA

When configured as an analog input, the following characteristics apply:

Input/output..... Input
Resolution..... 10 bits (0...1023 raw value)
V_{IN} range..... 0V to VCC
Input capacitance..... 7pF typical

When configured as a pulse output, the following characteristics apply:

Input/output..... Output
Voltage range ..... 0V to 3.3V
Waveform frequency..... 0.06 Hz - 6 MHz
Duty cycle range…..... 0.00% to 100.00%
Waveform polarity…..... High at start of period
Waveform alignment…..... Left aligned

When configured as a pulse counter, this pin may be configured in combination with another I/O function. When configured with an additional function, the characteristics for that function apply to this pin. Otherwise, the pin is internally pulled-up and the following characteristics apply:

Input/output..... Input
5V tolerant ..... Yes
V_{IL} range ..... -0.3V to +0.8V
V_{IH} range..... 2.0V to 5.5V

Pin #10 (GPIO_5/PWM_3)

Multi-function capable pin that can be configured with one or more of the following functions:

- General-purpose digital input
- General-purpose digital output
- Pulse output (filter to produce analog output)
- Pulse counter (may be configured in combination with other I/O functions)

When configured as a general-purpose digital input, this pin is internally pulled-up and the following characteristics apply:

Input/output..... Input
5V tolerant ..... Yes
V_{IL} range ..... -0.3V to +0.8V
V_{IH} range..... 2.0V to 5.5V

When configured as a general-purpose digital output, the following characteristics apply:

Input/output..... Output
V_{OL} (max)..... 0.4V
V_{OH} (min)..... VCC-0.4V
I_{OH} (max)..... 4mA
I_{OL} (max) ..... -4mA

When configured as a pulse output, the following characteristics apply:

Input/output..... Output
Voltage range ..... 0V to 3.3V
Waveform frequency..... 0.06 Hz - 6 MHz
Duty cycle range…..... 0.00% to 100.00%
Waveform polarity…..... High at start of period
Waveform alignment…..... Left aligned

When configured as a pulse counter, this pin may be configured in combination with another I/O function. When configured with an additional function, the characteristics for that function apply to this pin. Otherwise, the pin is internally pulled-up and the following characteristics apply:

Input/output..... Input
5V tolerant ..... Yes
V_{IL} range ..... -0.3V to +0.8V
V_{IH} range..... 2.0V to 5.5V

Pin #17 (GND)

Module ground reference.

9 Appendix A: Database Endianness

A key feature of the PicoPort is the ability to change the byte order storage scheme for data in the database between big endian and little endian. The database endianness is the convention used to store multi-byte data to or retrieve multi-byte data from the database. The selected endianness affects the end-to-end consistency of multi-byte data between the two networks on the gateway.

To better understand how this byte-ordering scheme works, the following explains how the device stores and retrieves multi-byte data to and from the database. Data is stored into the database starting at the low address and filled to higher addresses. The endianness determines whether the most-significant or least-significant bytes are stored first.

Let's look at some examples that demonstrate this. Figure 3 shows how the hex value 0x12345678 is stored into the database using a big endian byte order. Since the hex value 12 is the most significant byte, it is stored at address "a", the lowest address.

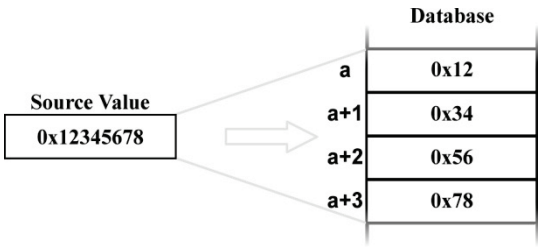


Figure 3: Big Endian Storage

Figure 4 demonstrates how the hex value 0x12345678 is stored into the database using a little endian byte order. Since the hex value 78 is the least significant byte, it is stored at the lowest address.

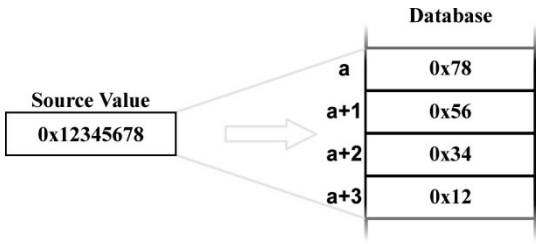


Figure 4: Little Endian Storage

Similarly, data is retrieved from the database starting at the low address. The endianness decides whether the first byte is interpreted as the least-significant byte or the most-significant byte of the multi-byte number.

Here are some examples that demonstrate this. Figure 5 shows how the hex value 0x12345678 is retrieved from the database using a big endian byte order. Since the hex value 12 is at address "a", the lowest address, it is the most significant byte.

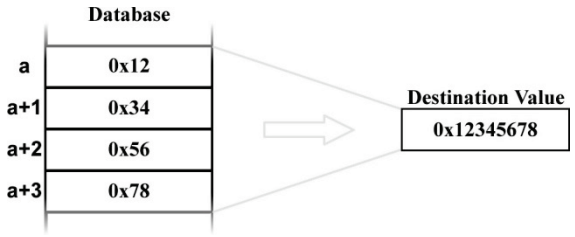


Figure 5: Big Endian Retrieval

Figure 6 demonstrates how the hex value 0x12345678 is retrieved from the database using a little endian byte order. Since the hex value 78 is at the lowest address, it is the least significant byte.

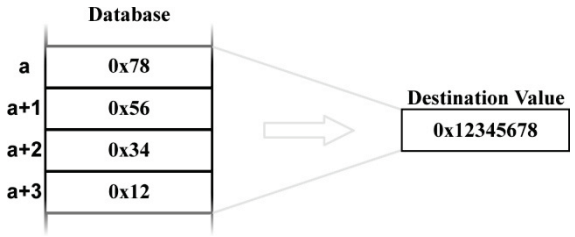


Figure 6: Little Endian Retrieval

The above examples illustrate the data movement to and from the device's internal database. This idea helps explain the data movement, as a whole, from one port to the other on the device between two different networks. Because networks vary in the manner that they exchange data, endianness selection must be part of the device's configuration in order to ensure coherent multi-byte data exchange. There are two data exchange methods used by the supported networks of the device.

The first method is used in those networks that define a byte order for how to interpret multi-byte data within an array of bytes. PROFIBUS, for example, defines a big-endian order for multi-byte data, while DeviceNet defines a little-endian order for multi-byte data. These networks exchange I/O data by means of a "bag of bytes" approach, whereas the device need not concern itself with where individual values are delimited within the array of bytes itself (as this is determined by the sending or receiving nodes on the networks). The bytes are simply stored into the database in the order they were received. Endianness selection therefore has no effect on data storage or retrieval with a "bag of bytes" protocol driver.

The other method is that used by networks that exchange data by means of an "object value" system, whereas data is exchanged by addressing a certain object to read or write data. Modbus for example, uses registers, while BACnet uses objects such as analog values to exchange data. When multi-byte values are received by the device, the bytes must be stored into the database in the order defined by the endianness selected. Likewise, when retrieving multi-byte values from the database for the device to transmit, the endianness selected will determine how the data is reconstructed when read from the database.

The selection of the correct byte ordering is crucial for coherent interaction between these two types of networks on the device. The following presents examples of how the database endianness affects end-to-end communication between networks and when each byte-ordering scheme should be used.

9.1 Modbus - PROFIBUS Example

This example shows the interaction between a network using an object value method (Modbus) and one using a bag of bytes method (PROFIBUS) to exchange data. The device reads holding registers 1 and 2 from the Modbus network, stores the data into the database, and then sends the 4 bytes of input data onto the PROFIBUS network. Figure 7 shows this data movement for the device's database configured as big endian. Because the PROFIBUS specification defines multi-byte values within the byte array to be interpreted as big endian, it is recommended that the database be configured for big-endian byte order when using PROFIBUS. In the example, holding register 1 has a value of 0x1234 and holding register 2 has a value of 0x5678. When the PROFIBUS device receiving the input data from the device recombines the two pairs of 2-byte values, the resulting data is 0x1234 and 0x5678, thus successfully receiving the correct values for holding registers 1 and 2.

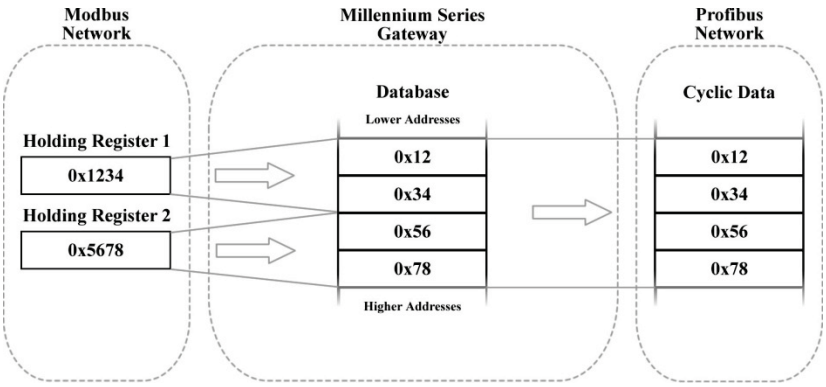


Figure 7: Modbus - PROFIBUS Big Endian

In contrast, Figure 8 shows the effects of configuring the database for little-endian byte order. Holding registers 1 and 2 again have values of 0x1234 and 0x5678, respectively. However, when the PROFIBUS device receiving the input data from the device interprets these values, the resulting pairs of 2-byte values become 0x3412 and 0x7856, thus receiving incorrect values for holding registers 1 and 2. Note that in both examples, the PROFIBUS network data is always identical, byte-for-byte, to the device's database. For this reason it is important to configure devices that use a bag-of-bytes style network, such as the PBDP-1000, to use the same endianness as defined for that network.

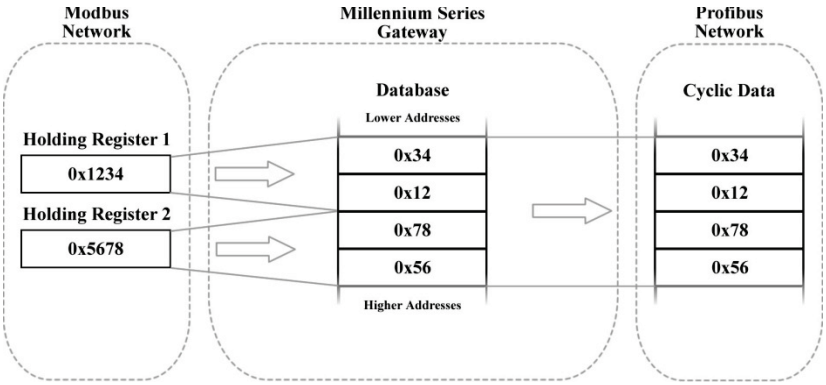


Figure 8: Modbus - PROFIBUS Little Endian

9.2 Modbus - DeviceNet Example

This example shows the interaction between a network using an object value method (Modbus) and one using a bag of bytes method (DeviceNet) to exchange data. The device reads holding registers 1 and 2 from the Modbus network, stores the data into the database, and then sends the 4 bytes of input data onto the DeviceNet network. Figure 9 shows this data movement for the device's database configured as little endian. Because the DeviceNet specification defines multi-byte values within the byte array to be interpreted as little endian, it is recommended that the database be configured for little-endian byte order when using DeviceNet. In the example, holding register 1 has a value of 0x1234 and holding register 2 has a value of 0x5678. When the DeviceNet device receiving the input data from the device recombines the two pairs of 2-byte values, the resulting data is 0x1234 and 0x5678, thus successfully receiving the correct values for holding registers 1 and 2.

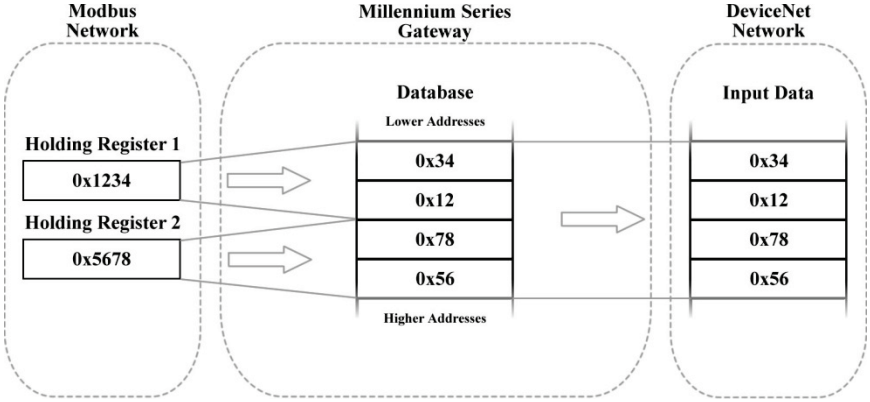


Figure 9: Modbus - DeviceNet Little Endian

In contrast, Figure 10 shows the effects of configuring the database for big-endian byte order. Holding registers 1 and 2 again have values of 0x1234 and 0x5678, respectively. However, when the DeviceNet device receiving the input data from the device interprets these values, the resulting pairs of 2-byte values become 0x3412 and 0x7856, thus receiving incorrect values for holding registers 1 and 2. Note that in both examples, the DeviceNet network data is always identical, byte-for-byte, to the device's database. For this reason it is important to configure devices that use a bag-of-bytes style network, such as the DNET-1000, to use the same endianness as defined for that network.

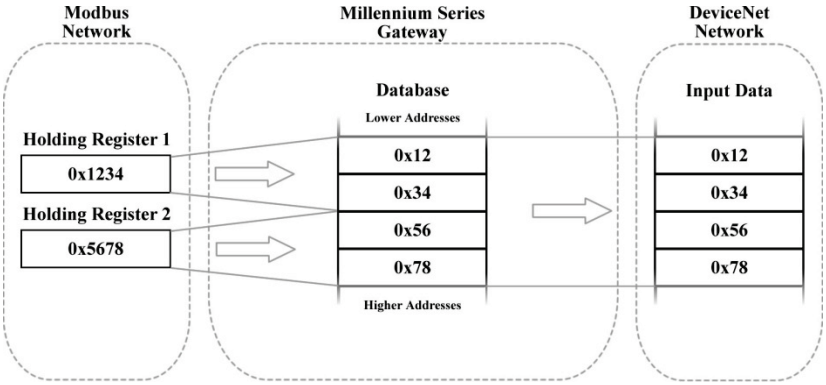


Figure 10: Modbus - DeviceNet Big Endian

9.3 BACnet - DeviceNet Example

This example is quite similar to the previous one as data is exchanged between an object-value style network (BACnet) and a bag-of-bytes style network (DeviceNet). The key difference is that in this example, BACnet Analog Value 0 is a 32-bit value, as opposed to two 16-bit Modbus registers. Here, the device reads analog value 0 from the BACnet network, stores the data into the database, and sends the input data onto the DeviceNet network. Figure 11 demonstrates the data flow from the BACnet network to the DeviceNet network through a device configured to use a little endian database. Because the DeviceNet specification defines multi-byte values within the byte array to be interpreted as little endian, it is recommended that the database be configured for little-endian byte order when using DeviceNet. In the example, analog value 0 has a value of 0x12345678. When the DeviceNet device receiving the input data from the device interprets the 4 bytes, the resulting 4-byte value will be 0x12345678, thus successfully receiving the original value of the BACnet analog value object.

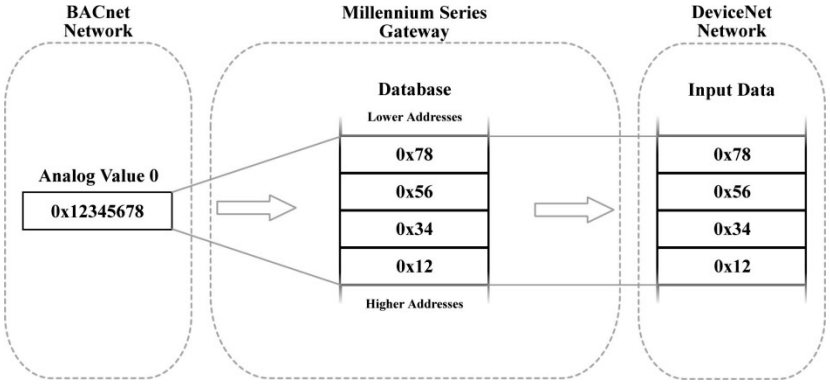


Figure 11: BACnet - DeviceNet Little Endian

Conversely, Figure 12 illustrates the consequences of configuring the database for big-endian byte order using this scenario. Once again, Analog Value 0 has a value of 0x12345678. But now, when the DeviceNet device interprets the 4 bytes of input data sent by the device, the resulting 4-byte value is 0x78563412, thus receiving an incorrect value for Analog Value 0. Note that in this example as well, the DeviceNet byte array is identical, byte-for-byte to the database. This example, in conjunction with the previous, demonstrates the dependence on the bag-of-bytes style networks for correct database endianness selection.

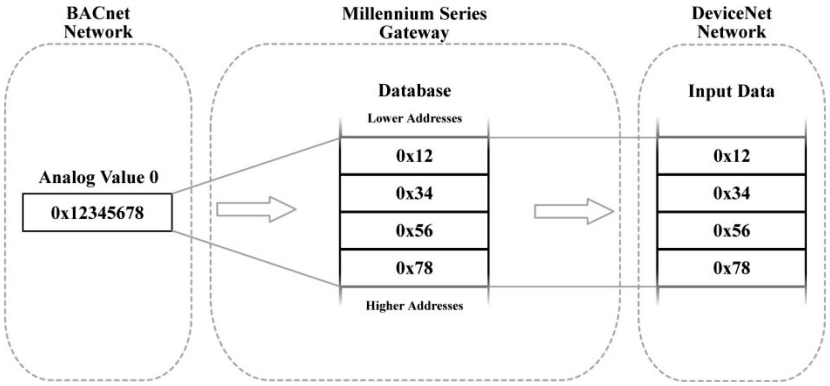


Figure 12: BACnet - DeviceNet Big Endian

9.4 BACnet - Modbus Analog Element Example

This example exhibits two networks that both use an object value scheme to exchange data. In this scenario, the database endianness is irrelevant if the data types are the same for both networks. This example shows communication between a BACnet network and a Modbus network using two 16-bit analog value BACnet objects and two 16-bit Modbus holding registers. As shown in Figure 13, the values from the BACnet network are stored into the database with big-endian byte ordering. Figure 14 shows the values from the BACnet network being stored into the database with little-endian byte ordering. Regardless of the byte-ordering scheme used, the two holding registers on the Modbus network receive the same values. Notice that in both cases, analog values 1 and 2 have values of 0x1234 and 0x5678, respectively, while holding registers 1 and 2 also have values of 0x1234 and 0x5678, respectively. The only difference between the two cases is how the data is being stored internally on the device itself.

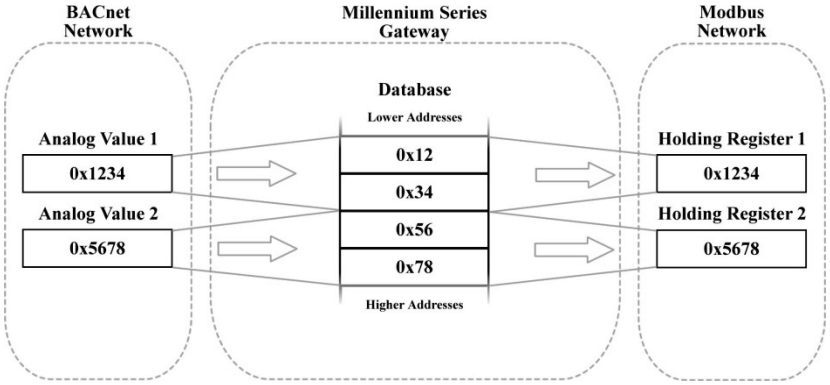


Figure 13: BACnet - Modbus (Analog Objects & Registers) Big Endian

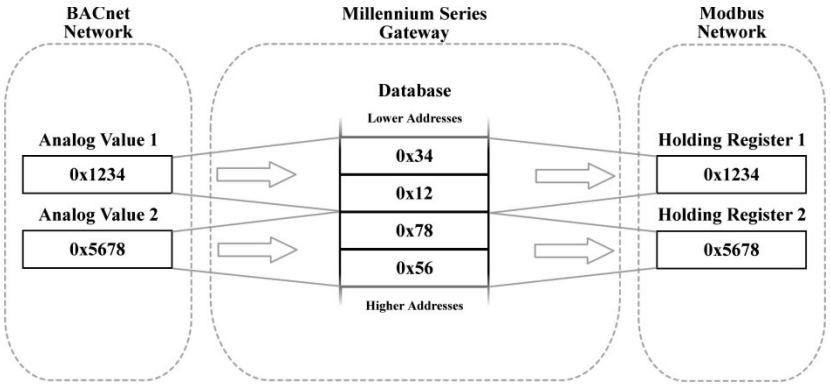


Figure 14: BACnet - Modbus (Analog Objects & Registers) Little Endian

9.5 BACnet - Modbus Binary Element Example

This example also contains two networks that both employ an object value method for exchanging data, but unlike the previous example, the database endianness does affect the end-to-end alignment of the data. In this example, communication is taking place between a BACnet network and a Modbus network using single-bit data elements. The BACnet side is using binary values 1 through 32, while the Modbus side is using coil status 1 through 32. The byte ordering of the database is significant because of the manner in which Modbus coils are mapped in the device. Coils (and input statuses) are mapped to registers, not addresses (refer to the Modbus driver documentation for more information). Since registers are 16-bit entities, the byte order of the registers (and by association, the coils), is affected by the endianness configured for the database. BACnet binary objects, however, are mapped on a byte-wise basis into the database.

When the database is configured for a little-endian byte order, binary values 1…8 corresponds to coils 1…8, binary values 9…16 corresponds to coils 9…16, and so on. This can be seen in Figure 15. Notice that the least significant bytes of the registers that the coils map to are placed in the lower memory addresses in the database. Because Modbus discretes are mapped to registers in a bit-wise little-endian fashion, it is recommended that the database be little endian in this scenario so that bit-wise data will align between networks.

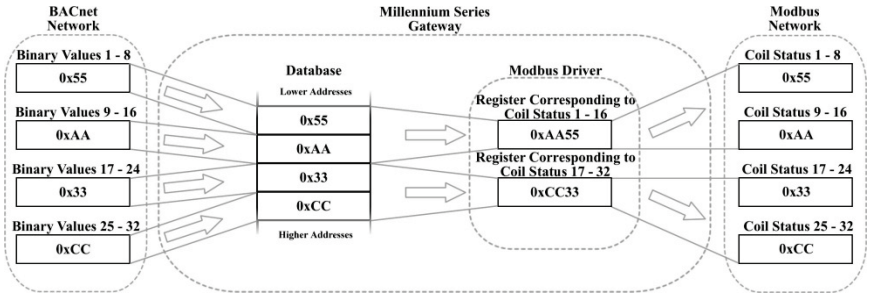


Figure 15: BACnet - Modbus (Binary Objects & Discretes) Little Endian

However, when the database is configured for a big-endian byte order, binary values 1…8 correspond to coils 9…16, binary values 9…16 correspond to coils 1…8, and so on. This can be seen in Figure 16. Since the most significant bytes of the Modbus registers that the coils map to are now mapped to lower addresses, the alignment between the two networks' bit-wise data is

byte swapped. While this alignment can still be used, it is much more intuitive when the database is configured to be little endian.

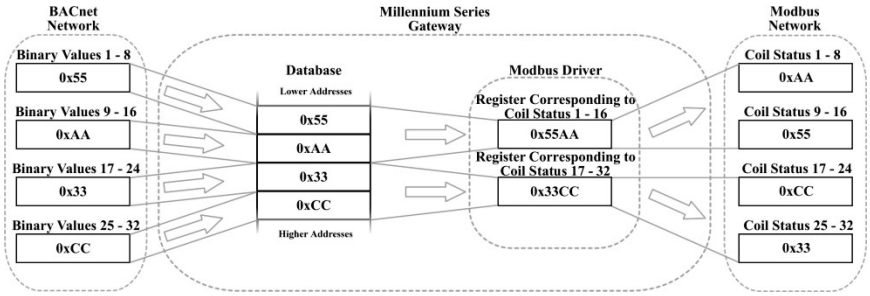


Figure 16: BACnet - Modbus (Binary Objects & Discretes) Big Endian

10 Appendix B: Diagnostics Objects

This section details the information that is enabled by adding a diagnostics object to a service object. Figure 17 diagrams the structure of this diagnostics information. Because this 16-byte structure resides in the database at a user-designated location, it can be accessed from any supported network or protocol in order to continuously determine the health and performance of the corresponding service object.

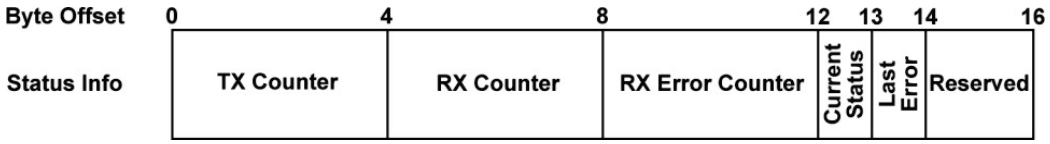


Figure 17: Diagnostics Object Format

TX Counter

A 32-bit counter that increments when the driver transmits a packet.

RX Counter

A 32-bit counter that increments when the driver receives a valid packet.

RX Error Counter

A 32-bit counter that increments when the device receives an error response packet, or when an error occurs upon reception of a packet.

Current Status

Indicates the status of the most-recently received packet. This field is updated each time the “RX Counter” or “RX Error Counter” increments. Refer to Table 25 for a list of supported codes.

Last Error

Indicates the last reception error that occurred. This field is updated each time the “RX Error Counter” increments. Refer to Table 25 for a list of supported codes.

Reserved

These two bytes are reserved for future use.

Table 25: Status / Error Codes

Status / Error Code (Hex)	Description
0x00	No Error
0xF0	Invalid Data Address
0xF1	Data Error
0xF2	Write To Read-Only
0xF3	Read From Write-Only
0xF4	Target Busy
0xF5	Target Error
0xF6	Cannot Execute
0xF7	Mode Error
0xF8	Other Error
0xF9	Memory Error
0xFA	Receive Error
0xFB	Invalid Function
0xFC	Invalid Packet
0xFD	Security Error
0xFE	Checksum Error
0xFF	Timeout Error

11 Appendix C: BACnet PICS

BACnet Protocol Implementation Conformance Statement (PICS)

Date:	December 12, 2016
Vendor Name:	ICC, Inc.
Product Name:	PicoPort Communications Module
Product Model Number:	PicoPort
Applications Software Version:	V2.500
Firmware Revision:	V2.500
BACnet Protocol Revision:	12

Product Description:

The PicoPort is a miniature serial communications engine-on-module for OEM applications. This product supports native BACnet, connecting directly to the MS/TP LAN using baud rates of 9600, 19200, 38400, 57600, 76800, and 115200. The device can be configured as a BACnet Client or as a BACnet Server.

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-A (DS-RP-A)
☒ Data Sharing – ReadProperty-B (DS-RP-B)
☒ Data Sharing – ReadPropertyMultiple-B (DS-RPM-B)
☒ Data Sharing – WriteProperty-A (DS-WP-A)
☒ 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-A (DM-DDB-A)
☒ 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)
☒ Device Management – TimeSynchronization-B (DM-TS-B)*
☒ Device Management – UTCTimeSynchronization-B (DM-UTC-B)*

* Available only when Real-time Clock Settings are enabled

Segmentation Capability:

☐ Able to transmit segmented messages Window Size _____
☐ Able to receive segmented messages Window Size _____

Standard Object Types Supported:

Property	Object Type									
	Device	Binary Input	Binary Output	Binary Value	Analog Input	Analog Output	Analog Value	Multi-state Input	Multi-state Output	Multi-state Value
Object Identifier	R	R	R	R	R	R	R	R	R	R
Object Name	R	R	R	R	R	R	R	R	R	R
Object Type	R	R	R	R	R	R	R	R	R	R
System Status	R									
Vendor Name	R									
Vendor Identifier	R									
Model Name	R									
Firmware Revision	R									
App Software Revision	R									
Protocol Version	R									
Protocol Revision	R									
Services Supported	R									
Object Types Supported	R									
Object List	R									
Max APDU Length	R									
Segmentation Support	R									
Local Time*	R									
Local Date*	R									
UTC Offset*	W (-840…720)									
Daylight Savings Status*	W									
APDU Timeout	W (10…65535)									
Number APDU Retries	W (0…10)									
Max Master	W (1…127)									
Max Info Frames	R									
Device Address Binding	R									
Database Revision	R									
Active COV Subscriptions	R									
Present Value		R	W	W	R	W	W	R	W	W
Status Flags		R	R	R	R	R	R	R	R	R
Event State		R	R	R	R	R	R	R	R	R
Reliability		R	R	R	R	R	R	R	R	R
Out-of-Service		R	R	R	R	R	R	R	R	R
Number of States								R	R	R
Units					R	R	R			

Priority Array			R	R		R	R		R	R
Relinquish Default			R	R		R	R		R	R
COV Increment					W	W	W			
Polarity		R	R							
Inactive Text		R	R	R						
Active Text		R	R	R						

R – Readable using BACnet services
W – Readable and writable using BACnet services

***** Available only when Real-time Clock Settings are enabled

Data Link Layer Options:

☐ BACnet IP, (Annex J)
☐ BACnet IP, (Annex J), Foreign Device
☐ ISO 8802-3, Ethernet (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

Network 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.

☒ ISO 10646 (UTF-8) ☐ IBM™/Microsoft™ DBCS
☐ JIS X 0208 ☐ ISO 10646 (UCS-4) ☐ ISO 10646 (UCS-2) ☐ ISO 8859-1

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

Refer to protocol-specific manuals for other supported protocols.

ICC

INDUSTRIAL CONTROL COMMUNICATIONS, INC.

1600 Aspen Commons, Suite 210
Middleton, WI USA 53562-4720
Tel: [608] 831-1255 Fax: [608] 831-2045

http://www.iccdesigns.com

Printed in U.S.A