

PRELIMINARY

# Babel Buster 485XM User Guide



**Control Solutions, Inc.  
2179 Fourth Street  
White Bear Lake, MN 55110**

***www.csimn.com***

PRELIMINARY

Control Solutions, Inc.  
*Babel Buster® 485XM*  
Modbus RTU to BACnet MS/TP Gateway

User Guide  
Rev. 1.01 • June 2007

This manual corresponds with firmware version 1.01.

## IMPORTANT SAFETY CONSIDERATIONS:

Proper system design is required for reliable and safe operation of distributed control systems incorporating Babel Buster series gateways and other such devices. It is extremely important for the user and system designer to consider the effects of loss of power, loss of communications, and failure of components in the design of any monitoring or control application. This is especially important where the potential for property damage, personal injury, or loss of life may exist. By using the Babel Buster series gateway or any other Control Solutions, Inc., product, the user has agreed to assume all risk and responsibility for proper system design as well as any consequence for improper system design.

© 2007 Control Solutions, Inc.

BACnet® is a registered trademark of American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE). Babel Buster® is a registered trademark of Control Solutions, Inc., Minnesota, USA. All other trademarks mentioned in this document are the property of their respective owners. Information in this document is subject to change without notice and does not represent a commitment on the part of Control Solutions, Inc. This document is provided “as is,” without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of fitness or merchantability for a particular purpose. Control Solutions may make improvements and/or changes in this manual or in the product(s) and/or the program(s) described in this manual at any time. This product could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes may be incorporated in new editions of the publication.

## Babel Buster 485XM Overview

The Babel Buster 485XM BACnet Gateway supports connecting a Modbus RTU device to a BACnet MS/TP network. It does not support connecting a BACnet device to a Modbus network. The 485XM gateway expects to be the Modbus master, and cannot be configured to operate as a slave. A fixed number of Modbus devices, with slave ID's or unit numbers starting at one, is supported. The 485XM gateway is a BACnet slave (server) only, and does not include a BACnet client.

There is a one to one mapping of Modbus registers to BACnet objects. The definition of "register" for purposes of defining this one to one relationship is any standard Modbus register, namely: coil, discrete input, input register, or holding register. Therefore a mapping relationship may refer to one bit (coil or discrete input) or one 16-bit register (input register or holding register). There is a special case where a "register" actually consists of two consecutive Modbus registers: When the register format is defined as "double integer" or "floating point", two consecutive registers are treated as one data element. Floating point is IEEE 754 (big endian).

Use an Analog Input object to read a Modbus input register or holding register. You can use an Analog Input to read Modbus coils or discrete inputs; however, the functionality will then be equivalent to a Binary Input. If the Analog Input object is placed out of service, it can be used to write to a Modbus holding register.

Use an Analog Value object to provide the most effective read/write access to a Modbus holding register. The gateway will periodically read the Modbus register, and its contents will be accessible as the Present Value of the BACnet object. A write to Present Value requires a priority level, and this value is placed into the object's priority array. The result of processing the priority array according to BACnet standard will be written to the Modbus holding register any time the Present Value is written. To reiterate, the highest priority value is written to the Modbus holding register, and the contents actually read back from the holding register are returned as the Present Value read from the BACnet object. In other words, if the Modbus device is not behaving properly, the value written will not be the value read.

Use a Binary Input to read Modbus coils and discrete inputs. A Binary Input can be used to read a holding register, but the result will only indicate whether the register contained a nonzero value. If the Binary Input object is placed out of service, it can be used to write to a Modbus coil.

Use a Binary Value object to provide the most effective read/write access to a Modbus coil. The Binary Value behaves the same as Analog Value, except that the Binary Value applies to single bit objects, namely the Modbus coil. The Binary Value's use of the priority array is identical to use in the Analog Value.

Communication problems between the gateway and Modbus device will be logged and reported via the Reliability property of the affected object. Vendor specific Reliability codes indicate "no response", "exception", etc. If the object is in fault as a result of communication problems, the Status Flags will indicate Fault, and the Event State will

also indicate Fault. In addition, error counts per Modbus slave ID may be read via the BACnet Device object in the gateway.

**Babel Buster 485XM Capacities:**

Modbus slave devices: 25

Analog Input objects: 40

Analog Value objects: 10

Binary Input objects: 40

Binary Value objects: 10

### **Initial Setup or Setup Recovery**

The gateway's station ID, device ID, and baud rate for the BACnet MS/TP port are set by writing properties to the device object. This of course presumes that you know what they are initially. By holding the INIT/ATN button down during power-up, you force the station ID to one (1), device ID to one (1), and baud rate to 9600. It is assumed that you will make these one-time device settings while the gateway is not connected to the rest of your BACnet network, and is instead connected to a PC running BACbeat or Control Solutions proprietary Babel Buster BAConfig tool.

### **General Discussion of Object Behavior**

#### **Writeable Input Objects**

Input objects are normally read-only. They become write-only objects if placed out of service. Normal operation performs periodic Modbus polls to read the configured register, and place the contents in the Present Value buffer of the BACnet object. The optional slope and intercept may be used to scale the raw Modbus data to more desirable units in the BACnet object.

Writes to an Input object will normally not be accepted. However, if the object is placed out of service, you may now write the Present Value. The value written will be scaled using the optional slope and intercept, and periodically written to the configured Modbus register. While out of service, reading the BACnet Present Value will simply echo what was written. To determine if the contents were successfully written to the Modbus device, you need to check the Reliability property.

When transitioning an Input object to out-of-service, causing it to become a write-only object, the most recently read Modbus register contents will initially be written back to that register. Subsequent writes to the BACnet Present Value will write that new value (after optional scaling) to the Modbus register. When transitioning back to in-service, the BACnet Present Value will be immediately replaced with whatever is read from Modbus.

## Value Objects

Value objects are read-write objects, and used with Modbus registers that are also read-write (either holding registers or coils). Value objects provide full priority array functionality in controlling the contents of the Modbus register. A priority level must be included when writing the Present Value of a Value object. The highest priority level not yet relinquished will be written to the Modbus register each time the Present Value is updated. If all priorities are relinquished, the relinquish default value is written to the Modbus register when the last level is relinquished. The Modbus register contents read back will be placed in the value object's Present Value buffer for subsequent BACnet reading. The optional slope and intercept are applied, scaling Modbus data to BACnet data. If the object is placed out of service, the Present Value may be written directly, and the result is a direct write to the Modbus register. Modbus is treated as write-only when the Value object is out-of-service.

Upon transitioning a Value object to out-of-service, the most recent priority based update will remain in effect in the Modbus register until the next time Present Value is written. The new value will be written to Modbus at that time. While out-of-service, values may continue to be written to the priority array by writing Present Value with a priority. You must write Present Value with no priority to do a direct write while out-of-service (writes with priority will simply be placed in the priority array without writing to Modbus). Upon transition to in-service, the highest priority non-relinquished value found in the priority array will be written to the Modbus register.

## Sequences of Modbus Registers

Ideally, when sequential multiple Modbus registers are read, they will be read in a single request to reduce traffic on the communication link and improve response time of the overall system. If sequential contiguous BACnet objects are configured to access sequential contiguous Modbus registers of the same type in the same device, they will be combined into a single Modbus request to the respective device. The response will be distributed to the multiple BACnet objects affected. It should be noted that if the request fails, all BACnet objects contained in the request will be flagged as being in fault even if only one of the multiple Modbus registers caused an exception. To avoid combining requests, simply define the objects out of order.

## Packed Modbus Registers

Packed registers are a special case in which multiple BACnet objects are mapped to a single Modbus register. It is common to find a VFD, for example, which uses a single holding register which includes a bit for start/stop, a bit for forward/reverse, bits for other functions, and possibly an 8-bit field for a function code along with the other bits. The Babel Buster 485XM allows you to simplify your BACnet application by parsing these packed registers for you when read, and formatting them from multiple objects when written.

Packed registers will not be included in a combined sequence as mentioned above. Furthermore, packed registers do not have to be defined in contiguous BACnet objects,

and do not even have to be of the same BACnet object type. You signal your desire for a packed register to the gateway by setting the Object\_Map\_Use property to “grouped” rather than “normal”. When processing a grouped object, the gateway will search through the entire array of objects (all types, analog and binary) looking for all objects that map to the same Modbus register in the same Modbus device.

BACnet object data may translate into a single bit in a holding register, or a bit field of 2 to 16 bits. Packed registers apply only to holding registers, and only to unsigned 16-bit integer data. Floating point data may not be processed by a “grouped” object. Slope and intercept are ignored when Object\_Map\_Use is set to “grouped”. However, two other properties, Register\_Bit\_Mask and Register\_Bit\_Fill are *only* used when the object is “grouped”.

The bit mask defines which bits in the Modbus register are to be singled out for interpretation by the BACnet object. BACnet data is pulled from those bit positions and right justified when reading. BACnet data is shifted left and stuffed into those bit positions when writing. The 16-bit “fill” value is bitwise OR’ed into the data after all bit fields have been assembled for writing to Modbus, and can be used to see that a ‘1’ bit is always written to a certain position. The bit fill is not used when reading.

It should be noted that the transition from in service, to out of service, and back, is not as “clean” for packed registers. This is due to the fact that the single register, during transition, will be combining both in-service and out-of-service BACnet objects. You should anticipate needing to re-write the Present Value all BACnet objects (mapped to the affected packed register) after service transitions are complete, and also anticipate that a value of zero may be written to some bit positions of the Modbus object while BACnet objects are in transition. If you are using input objects to write Modbus registers, it is suggested that you use the Special\_Service\_Request bit to default to out of service at power-up, thus eliminating the need for transitions.

The properties that are recognized for each type of object supported in Babel Buster 485XM are listed below, by object type. The BACnet property name is given in the left column, and its property ID value is in parenthesis after the name. All listed properties are readable, and those having (W) after the property ID are writeable. The property type is listed in the right column, including applicable enumeration values.

**ANALOG INPUT**

Object_Identifier (75)	BACnetObjectIdentifier
Object_Name (77)	CharacterString “Analog Input <i>n</i> ”
Object_Type (79)	BACnetObjectType ENUMERATED: analog-input (0) analog-value (2) binary-input (3) binary-value (5) device (8)
Present_Value (85) (W)	REAL (no index, no priority) (writeable only when out of service)
Status_Flags (111)	BACnetStatusFlags BIT STRING: fault(1), out-of-service(3)
Event_State (36)	BACnetEventState ENUMERATED: normal(0), fault(1)
Reliability (103)	BACnetReliability ENUMERATED: normal(0) <i>Vendor specific:</i> no response (64) crc error (65) exception, illegal function code (66) exception, illegal data address (67) exception, illegal data value (68) exception, code+65, rarely used (69..79) configuration property fault (80)
Out_Of_Service (81) (W)	BOOLEAN
Units (117)	BACnetEngineeringUnits
<u><i>Vendor Specific Object Properties:</i></u>	
Object_Map_Use (601) (W)	ENUMERATED: unused(0), normal(1), grouped(2)
Special_Service_Request (602) (W)	BIT STRING: (1) default to out of service at power-up (2) invoke write at power-up if writeable object (3) set BACnet Present Value to default if Modbus communications fails

PRELIMINARY

Scan_Period (603) (W)	Unsigned Modbus poll/update time in seconds
Register (604) (W)	Unsigned Modbus register number 1..65535
Register_Type (605) (W)	ENUMERATED: none(0), coil(1), discrete-input(2), input-register(3), holding-register(4)
Register_Format (606) (W)	ENUMERATED: none(0), signed-integer(1), unsigned-integer(2), double-signed-integer(3), double-unsigned-integer(4), floating-point(5), bit(6)
Modbus_Slave_ID (607) (W)	ENUMERATED: 1..MAX_RTU_DEVICE
Register_Bit_Mask (608) (W)	Unsigned
Register_Bit_Fill (609) (W)	Unsigned
Slope (610) (W)	REAL BACnet = Modbus * slope + intercept Modbus = (BACnet – intercept) / slope
Intercept (611) (W)	REAL
Default_Value (612) (W)	REAL
Reference_String (613) (W)	CharacterString

**ANALOG VALUE**

Object_Identifier (75)	BACnetObjectIdentifier
Object_Name (77)	CharacterString “Analog Value <i>n</i> ”
Object_Type (79)	BACnetObjectType ENUMERATED: analog-input (0) analog-value (2) binary-input (3) binary-value (5) device (8)
Present_Value (85) (W)	REAL (no index)
Status_Flags (111)	BACnetStatusFlags BIT STRING: fault(1), out-of-service(3)
Event_State (36)	BACnetEventState ENUMERATED: normal(0), fault(1)
Reliability (103)	BACnetReliability ENUMERATED: normal(0) <i>Vendor specific:</i> no response (64) crc error (65) exception, illegal function code (66) exception, illegal data address (67) exception, illegal data value (68) exception, code+65, rarely used (69..79) configuration property fault (80)
Out_Of_Service (81) (W)	BOOLEAN
Priority_Array (87)	BACnetPriorityArray SEQUENCE SIZE (16) OF BACnetPriorityValue REAL (each element)
Relinquish_Default (104) (W)	REAL
Units (117)	BACnetEngineeringUnits
<u><i>Vendor Specific Object Properties:</i></u>	
Object_Map_Use (601) (W)	ENUMERATED: unused(0), normal(1), grouped(2)

PRELIMINARY

Special_Service_Request (602) (W)	BIT STRING: (1) default to out of service at power-up (2) invoke write at power-up if writeable object (3) set BACnet Present Value to default if Modbus communications fails
Scan_Period (603) (W)	Unsigned Modbus poll/update time in seconds
Register (604) (W)	Unsigned Modbus register number 1..65535
Register_Type (605) (W)	ENUMERATED: none(0), coil(1), discrete-input(2), input-register(3), holding-register(4)
Register_Format (606) (W)	ENUMERATED: none(0), signed-integer(1), unsigned-integer(2), double-signed-integer(3), double-unsigned-integer(4), floating-point(5), bit(6)
Modbus_Slave_ID (607) (W)	ENUMERATED: 1..MAX_RTU_DEVICE
Register_Bit_Mask (608) (W)	Unsigned
Register_Bit_Fill (609) (W)	Unsigned
Slope (610) (W)	REAL $BACnet = Modbus * slope + intercept$ $Modbus = (BACnet - intercept) / slope$
Intercept (611) (W)	REAL
Default_Value (612) (W)	REAL
Reference_String (613) (W)	CharacterString

**BINARY INPUT**

Object_Identifier (75)	BACnetObjectIdentifier
Object_Name (77)	CharacterString “Binary Input <i>n</i> ”
Object_Type (79)	BACnetObjectType ENUMERATED: analog-input (0) analog-value (2) binary-input (3) binary-value (5) device (8)
Present_Value (85) (W)	BOOLEAN (no index, no priority) (writeable only when out of service)
Status_Flags (111)	BACnetStatusFlags BIT STRING: fault(1), out-of-service(3)
Event_State (36)	BACnetEventState ENUMERATED: normal(0), fault(1)
Reliability (103)	BACnetReliability ENUMERATED: normal(0) <i>Vendor specific:</i> no response (64) crc error (65) exception, illegal function code (66) exception, illegal data address (67) exception, illegal data value (68) exception, code+65, rarely used (69..79) configuration property fault (80)
Out_Of_Service (81) (W)	BOOLEAN
Polarity (84)	BACnetPolarity ENUMERATED: normal(0)
Units (117)	BACnetEngineeringUnits
<u><i>Vendor Specific Object Properties:</i></u>	
Object_Map_Use (601) (W)	ENUMERATED: unused(0), normal(1), grouped(2)
Special_Service_Request (602) (W)	BIT STRING: (1) default to out of service at power-up

PRELIMINARY

(2) invoke write at power-up if writeable object  
(3) set BACnet Present Value to default if  
Modbus communications fails

Scan_Period (603) (W)	Unsigned Modbus poll/update time in seconds
Register (604) (W)	Unsigned Modbus register number 1..65535
Register_Type (605) (W)	ENUMERATED: none(0), coil(1), discrete-input(2), input-register(3), holding-register(4)
Register_Format (606) (W)	ENUMERATED: none(0), signed-integer(1), unsigned-integer(2), double-signed-integer(3), double-unsigned-integer(4), floating-point(5), bit(6)
Modbus_Slave_ID (607) (W)	ENUMERATED: 1..MAX_RTU_DEVICE
Register_Bit_Mask (608) (W)	Unsigned
Register_Bit_Fill (609) (W)	Unsigned
Slope (610) (W)	REAL $BACnet = Modbus * slope + intercept$ $Modbus = (BACnet - intercept) / slope$
Intercept (611) (W)	REAL
Default_Value (612) (W)	REAL
Reference_String (613) (W)	CharacterString

**BINARY VALUE**

Object_Identifier (75)	BACnetObjectIdentifier
Object_Name (77)	CharacterString “Binary Value <i>n</i> ”
Object_Type (79)	BACnetObjectType ENUMERATED: analog-input (0) analog-value (2) binary-input (3) binary-value (5) device (8)
Present_Value (85) (W)	BOOLEAN (no index)
Status_Flags (111)	BACnetStatusFlags BIT STRING: fault(1), out-of-service(3)
Event_State (36)	BACnetEventState ENUMERATED: normal(0), fault(1)
Reliability (103)	BACnetReliability ENUMERATED: normal(0) <i>Vendor specific:</i> no response (64) crc error (65) exception, illegal function code (66) exception, illegal data address (67) exception, illegal data value (68) exception, code+65, rarely used (69..79) configuration property fault (80)
Out_Of_Service (81) (W)	BOOLEAN
Priority_Array (87)	BACnetPriorityArray SEQUENCE SIZE (16) OF BACnetPriorityValue BOOLEAN (each element)
Relinquish_Default (104) (W)	BOOLEAN
Units (117)	BACnetEngineeringUnits
<u><i>Vendor Specific Object Properties:</i></u>	
Object_Map_Use (601) (W)	ENUMERATED: unused(0), normal(1), grouped(2)

PRELIMINARY

Special_Service_Request (602) (W)	BIT STRING: (1) default to out of service at power-up (2) invoke write at power-up if writeable object (3) set BACnet Present Value to default if Modbus communications fails
Scan_Period (603) (W)	Unsigned Modbus poll/update time in seconds
Register (604) (W)	Unsigned Modbus register number 1..65535
Register_Type (605) (W)	ENUMERATED: none(0), coil(1), discrete-input(2), input-register(3), holding-register(4)
Register_Format (606) (W)	ENUMERATED: none(0), signed-integer(1), unsigned-integer(2), double-signed-integer(3), double-unsigned-integer(4), floating-point(5), bit(6)
Modbus_Slave_ID (607) (W)	ENUMERATED: 1..MAX_RTU_DEVICE
Register_Bit_Mask (608) (W)	Unsigned
Register_Bit_Fill (609) (W)	Unsigned
Slope (610) (W)	REAL $BACnet = Modbus * slope + intercept$ $Modbus = (BACnet - intercept) / slope$
Intercept (611) (W)	REAL
Default_Value (612) (W)	REAL
Reference_String (613) (W)	CharacterString

**DEVICE**

Object_Identifier (75)	BACnetObjectIdentifier
Object_Name (77)	CharacterString
Object_Type (79)	BACnetObjectType
System_Status (112)	BACnetDeviceStatus
Vendor_Name (121)	CharacterString
Vendor_Identifier (120)	Unsigned16 (should always return 208)
Model_Name (70)	CharacterString
Firmware_Revision (44)	CharacterString
Application_Software_Version (12)	CharacterString
Protocol_Version (98)	Unsigned
Protocol_Revision (139)	Unsigned
Protocol_Services_Supported (97)	BACnetServicesSupported
Protocol_Object_Types_Supported (96)	BACnetObjectTypesSupported
Object_List (76)	BACnetARRAY[N] of BACnetObjectIdentifier
Max_APDU_Length_Accepted (62)	Unsigned
Segmentation_Supported (107)	BACnetSegmentation
APDU_Timeout (11)	Unsigned
Number_Of_APDU_Retries (73)	Unsigned
Device_Address_Binding (30)	List of BACnetAddressBinding
Database_Revision (155)	Unsigned
<i><u>Vendor Specific Object Properties:</u></i>	
RTU_Port_Baud_Rate (651) (W)	ENUMERATED: 4800(0), 9600(1), 19200(2), 38400(3)

PRELIMINARY

RTU_Parity (652) (W)	ENUMERATED: none(0), odd(1), even(2), none_2_stop_bits(3)
RTU_Slave_Timeout (653) (W)	Unsigned Index: 1..MAX_RTU_DEVICE
RTU_Slave_Swap_Flag (654) (W)	ENUMERATED: no_swap(0), swapped(1) Index: 1..MAX_RTU_DEVICE
RTU_Message_Count (661) (W*)	Unsigned Index: 1..MAX_RTU_DEVICE
RTU_Exception_Count (662) (W*)	Unsigned Index: 1..MAX_RTU_DEVICE
RTU_Bad_CRC_Count (663) (W*)	Unsigned Index: 1..MAX_RTU_DEVICE
RTU_No_Response_Count (664) (W*)	Unsigned Index: 1..MAX_RTU_DEVICE

W\* -- Registers are “writeable” for purposes of resetting error counts. Writing any value will reset count to zero. Total message count will stop at 65535. Error counts will stop at 255. Counting will resume when reset to zero.

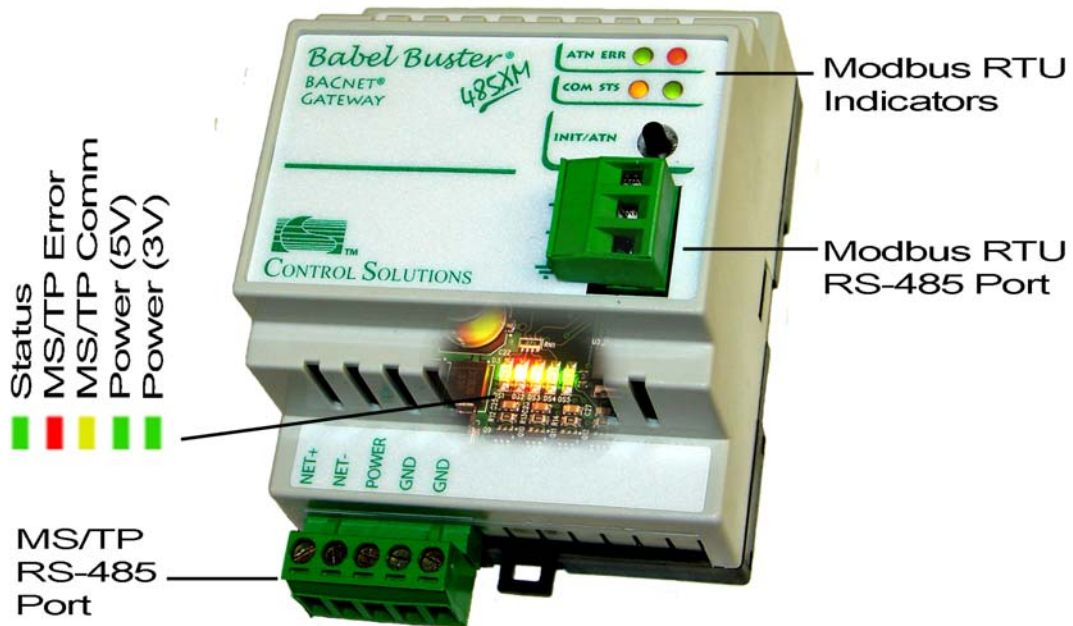
MSTP_Port_Baud_Rate (1201) (W)	ENUMERATED: 9600(0), 19200(1), 38400(2), 76800(3)
Station_ID (1202) (W)	Unsigned
Device_ID (1203) (W)	Unsigned
Max Masters (1204) (W)	Unsigned
Reinit_Password (1205) (WO)	CharacterString Note: This property is write-only, and is only writeable in INIT mode.

NOTE: Changes to port settings (properties 1201-1205) will take effect only upon issuing a Reinit Device command to the device. Upon receiving the Reinit Device, the gateway will commit these changes to EEPROM, and then reset itself so the new settings can take effect.

NOTE: The port settings will only be re-written if a WARM START command is issued. A cold start will only reset the device (same as power-up reset).

## INDICATORS

The LED indicators showing Modbus activity are on the front of the gateway. Additional diagnostic LEDs are inside the unit and may be viewed through the vent slots on the lower side.



### Modbus Indicators

ATN – Green, remains on blinking off briefly about once a second. This is the “I’m alive” indication. (This LED will also remain on while the INIT button is held down.)

COM – Yellow, flashes once each time a Modbus request is sent out.

STS – Green, flashes once each time a good Modbus response is received.

ERR – Red, flashes once each time a bad Modbus response is received, or no response is received (timed out). The definition of “bad” response is CRC error, or more commonly, an exception code was returned by the Modbus slave. Check the BACnet object reliability property to see what the error actually consisted of.

### BACnet and System Indicators

Status – Green, flashes each time the MS/TP token is passed. This is effectively a BACnet activity indicator.

MS/TP Error – Red, flashes each time a request is received for which an error is returned to the requesting client.

MS/TP Comm – Yellow, flashes each time a request is received for which a good response is returned to the requesting client.

Power – Green, remains on if power is present. The gateway base board has two power supplies, a 5V and a 3.3V supply. Only the 5V supply is used in this application, but the 3.3V indicator will still be present.

## WIRING

Connect the RS-485 network connections to the (+) and (-) terminals where shown. Connect power to the “power” terminal, and power common/ground to the “ground” terminal.

Power may be 10-30VDC, or 12-24VAC. This gateway uses a half wave rectifier which means it is not required to have a dedicated isolation transformer (ungrounded AC) as with other gateways in the Control Solutions gateway family that use a full wave bridge rectifier. There is no separate AC common on the Babel Buster 485XM. One side of AC will be grounded.

## ERROR CODES

Error codes returned via BACnet consist of an error class and code. The most common errors are listed below with class in parenthesis followed by code. Some applications will interpret these for you and provide a text description.

(property) code 9	Invalid data type
(property) code 32	Unknown property
(property) code 37	Value out of range
(property) code 40	Write access denied
(property) code 42	Invalid array index
(client) code 30	Timeout
(client) code 31	Unknown device
(object) code 31	Unknown object
(service) code 7	Inconsistent parameters
(service) code 10	Invalid access method
(service) code 29	Service request denied
(abort) code 4	Segmentation not supported
(reject) code 4	Invalid tag
(security) code 26	Password failure
(resource) code 0	Resource error – “other”