

AutomationDirect EBC I/O Device Driver Help

© 2009 Kepware Technologies

Table of Contents

1	Getting Started	4
	Help Contents	4
	Overview	4
2	Device Setup	4
	Device Setup	4
	Link Configuration	5
	Cable Diagrams	6
	Communication Settings	6
3	Performance Optimizations	7
	Optimizing Your AutomationDirect EBC Communications	7
4	Data Types Description	8
	Data Types Description	8
5	Address Descriptions	9
	Address Descriptions	9
	H2-EBC(-F) Addressing	9
	H4-EBC(-F) Addressing	9
	Terminator I/O Addressing	9
	GS Drive Addressing	10
	I/O Addressing	10
	H2-EBC(-F) I/O Addressing	10
	H4-EBC(-F) I/O Addressing	11
	Terminator I/O I/O Addressing	12
	Serial Port Addressing	14
	H2, H4, Terminator I/O Serial Port Addressing	14
	Serial Port Addresses	15
	Serial Data Received	15
	Number of RX Queue Bytes	15
	Flush Data Received	15
	Flush RX Queue	16
	Serial Data To Transmit	16
	Number of TX Queue Bytes	17
	Flush TX Queue	17
	Serial Port Configuration	17
	Baud Rate	17
	#Data Bits	18
	Parity	18
	#Stop Bits	18
	Mode	19
	Use RTS Line	19
	Pre-Transmit Delay	19
	Post-Transmit Delay	20
	GS Drive Addressing	20
	GS Drive Parameter Addressing	20
	GS Drive Status Addressing	21
6	Module Hot Swapping	22
	Module Hot Swapping	22

7 Error Descriptions.....	22
Error Descriptions.....	22
EBC Error Codes.....	24
Driver Error Messages.....	25
Driver Error Messages.....	25
Winsock initialization failed (OS Error = n).....	25
Winsock V1.1 or higher must be installed to use the AutomationDirect EBC device driver.....	26
Memory allocation error.....	26
Driver Warning Messages.....	26
Driver Warning Messages.....	26
Device '<device name>' not responding.....	26
Unable to write to '<address>' on device '<device name>'.....	27
Device address '<address>' contains a syntax error.....	27
Address '<address>' is out of range for the specified device or register.....	27
Device address '<address>' is not supported by model '<model name>'.....	27
Device address '<address>' is Read Only.....	27
Data type '<type>' is not valid for device address '<address>'.....	28
Device <device name> returned an error with value <error value> disabling Link Watchdog.....	28
Device <device name> returned an error with value <error value> enabling Link Watchdog.....	28
Read Errors	28
Read Errors.....	28
Address '<address>' is out of range for device '<device name>'.....	29
Base referenced in address '<address>' on device '<device name>' does not exist.....	29
Module referenced in address '<address>' on device '<device name>' does not exist.....	29
Address type invalid for address '<address>' on device '<device name>'.....	29
Device '<device name>' returned an error code with value.....	30
Device '<device name>' : Base '<base>' : Slot '<slot>' returned an error with value.....	30
Device '<device name>' : Base '<base>' : Slot '<slot>' returned a warning with value.....	30
Device '<device name>' : Base '<base>' : Slot '<slot>' returned information with value.....	31
Device '<device name>' : Base '<base>' : Slot '<slot>' returned an internal error with value.....	31
Device '<device name>' : Base '<base>' : Slot '<slot>' returned extended error.....	31
Frame received from device '<device name>' contains errors.....	32
Frame received from device '<device name>' contains errors. Verify its IP address.....	32
Format version of frame received from device '<device name>' is not supported.....	32
Model selected for device '<device name>' does not match the actual device model.....	32
Device '<device name>' block request [<start> to <end>] responded with exception <code>.....	33
Bad received length [<start> to <end>] on device '<device name>'.....	33
Write Errors	33
Write Errors.....	33
Write rejected. Address '<address>' is out of range for device '<device name>'.....	34
Write rejected. Base referenced in address '<address>' on device '<device name>'.....	34
Write rejected. Module referenced in address '<address>' on device '<device name>'.....	34
Write rejected. Invalid address type for address '<address>' on device '<device name>'.....	34
Write to address '<address>' failed. Device '<device name>' returned an undefined status code with value = '<value>'.....	34
Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned an error with value.....	35
Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned a warning with value.....	35
Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned information with value.....	35
Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned an internal error.....	36
Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned extended error.....	36

Write to address '<address>' failed. Frame received from device '<device name>' contains errors.....	36
Write to address '<address>' failed. Frame received from device '<device name>' contains errors.....	36
Write to address '<address>' failed. Format version of frame received from device '<device name>' is not supported.....	37
Write to address '<address>' failed. Model selected for device '<device name>' does not match the actual device model.....	37
Unable to write to address <address>. Device '<device name>' responded with exception <code>.....	37
8 Application Notes.....	37
Application Notes.....	37
Application Notes: T1F-14THM.....	38
Application Notes: D4-HSC.....	39
Index	41

AutomationDirect EBC I/O Device Driver Help

Help version 1.007

CONTENTS

[Overview](#)

What is the AutomationDirect EBC I/O Device Driver?

[Device Setup](#)

How do I configure a device for use with this driver?

[Optimizing Your AutomationDirect EBC Communications](#)

How do I get the best performance from the AutomationDirect EBC I/O driver?

[Data Types Description](#)

What data types are supported by this driver?

[Address Descriptions](#)

How do I address data locations?

[Module Hot Swapping](#)

How do I hot swap a module without "breaking" my client project?

[Error Descriptions](#)

What error messages are produced by the AutomationDirect EBC I/O driver?

[Application Notes](#)

Where can I find application notes for specific modules like the T1F-14THM?

Overview

The AutomationDirect EBC I/O Device Driver was designed specifically for use with 32 bit OPC server products running on Intel microprocessor based computers. It is intended for use with Koyo Direct/PLC Direct/AutomationDirect Logic Programmable Logic Controllers that may be accessed via an EBC or GS-EDRV Ethernet module. For operating system (OS) requirements, please refer to the OPC server help documentation.

Device Setup

Supported Devices

Terminator I/O,
H2-EBC, H2-EBC-F, H4-EBC, H4-EBC-F,
GS1, GS2 and GS3 Drives via GS-EDRV Ethernet module

Communication Protocol

Ethernet using Winsock V1.1 or higher.

Connection Timeout

This parameter specifies the time that the driver will wait for a connection to be made with a device. Depending on network load the connect time may vary with each connection attempt. The default setting is 3 seconds. The valid range is 1 to 60 seconds.

Request Timeout

This parameter specifies the time that the driver will wait on a response from the device before giving up and going on to the next request. Longer timeouts only affect performance if a device is not responding. The default setting is 250 milliseconds. The valid range is 50 to 9999 milliseconds.

Retry Attempts

This parameter specifies the number of times the driver will retry a message before giving up and going on to the next message. The default setting is 3 retries. The valid range is 1 to 10.

Device IDs

Up to 1024 devices may be defined on a given channel. Each device on the channel must be uniquely identified by its own IP address. The IP address is different from the Unit ID that is configurable on the I/O base unit. In general the device ID has the following format `YYY.YYY.YYY.YYY` where `YYY` designates the device IP address (each `YYY` byte should be in the range of 0 to 255).

The IP address for an EBC module can be determined using NetEdit, an AutomationDirect device configuration utility. NetEdit can be launched by selecting the Device ID Wizard button on the General tab of the Device Properties dialog:



NetEdit provides the ability to query the network, configure network parameters (i.e. IP Address), and update firmware for EBC devices.

Configuration

The H2-EBC(-F) and Terminator I/O EBC's automatically detect I/O modules in the base on power up. The H4-EBC(-F) however, only detects discrete modules. Use NetEdit to manually configure the H4-EBC(-F) base when using analog modules.

Link Configuration

Link Watchdog - EBC Only

The communications link between the PC and the EBC device is vital for the PC and EBC to exchange information. It is possible for this communication link to "break" during its use. A break in the communications link simply means there is a loss in communication or a lack of communication that is either permanent or temporary. An example of a loss in communication would be a physical break in the communication network. An example of a lack of communication would be a significant interval between PC-to-EBC transactions. This interval would have to exceed a set timeout before being considered a break in communication.

The client application can act on a loss of communication by continuously reading the `_Error` tag for the given EBC device. This is a `_System` level tag at the OPC server that is 0 when the link is good and 1 when the link is bad. Handling operations on the client side during a broken link is only half the story.

If the connection to the EBC is severed, it can no longer be given instructions for proper handling of I/O states such as digital outputs. There is no logic controller in an EBC system, therefore no ladder logic can be written to handle the situation. Fortunately there is a mechanism built into the EBC to safely handle link failure, called a Link Watchdog. When a specified amount of time elapses with a "broken" link, the watchdog wakes up and turns all device I/O outputs OFF.

The link watchdog, sometimes referred to as a link monitor, is configurable via the OPC server. It has two options: disable and enable.

Disable

When the link watchdog is disabled, a break in the communications link has no effect on the device's I/O outputs. The outputs will maintain the state they were assigned prior to the break until communications is restored and outputs are altered via the client application thereafter.

Enable

When enabled, the link watchdog continuously monitors the link. If a break in the link occurs, a timer is initiated. When the timer reaches the **Watchdog Timeout** value, all device I/O outputs will be turned OFF.

Watchdog Timeout

Values for the **Watchdog Timeout** range from 60ms to 32767ms.

Link Maintenance and Device Pings

For every request the device receives, the watchdog timer is reset. To prevent a watchdog timeout, at least one request must be made to the device before the watchdog timer expires. Normally the fastest tag scan rate would determine how often the device is communicated with. Such a dependency on tag scan rate would prove inflexible. To remedy this, each EBC device with the Link Monitor enabled will be pinged intermittently to ensure there are no unnecessary watchdog timeouts. Each ping request is sent independent of tag read/write requests. This removes the dependency from the tag scan rate for maintaining the link.

Link Monitor Ping = Low overhead request to the device with no response.

Link Monitor Ping Interval = **Watchdog Timeout** value / 3

Note: There are two factors that may impact how accurately the Link Monitor Ping request is sent at its scheduled interval: local PC usage and network activity.

Local PC Usage

For the following, the local PC is assumed to be the same PC running the OPC server in question. Since the OPC Server shares resources with other applications running on the local PC, there is no guarantee that each Link Monitor Ping request will be sent at its set interval. It is possible that these applications may preempt the OPC server long enough for a Link Monitor Ping request to be made after the watchdog timer expires.

Network Activity

The amount of activity on the network(s) providing access to the given EBC device can greatly affect the delivery of the Link Monitor Ping request. Collisions and load can delay a request long enough for a Link Monitor Ping request to reach the device after the watchdog timer expires.

Note: The Watchdog Timeout may have to be tuned accordingly in order to ensure spurious watchdog timeouts do not occur.

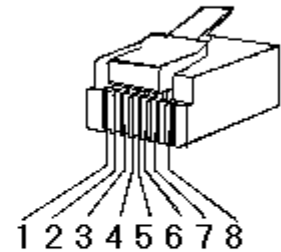
Cable Diagrams

Patch Cable (Straight Through)

TD + 1	OR/WHT	OR/WHT	1	TD +
TD - 2	OR	OR	2	TD -
RD + 3	GRN/WHT	GRN/WHT	3	RD +
4	BLU	BLU	4	
5	BLU/WHT	BLU/WHT	5	
RD - 6	GRN	GRN	6	RD -
7	BRN/WHT	BRN/WHT	7	
8	BRN	BRN	8	

RJ45 RJ45

10 BaseT



Crossover Cable

TD + 1	OR/WHT	GRN/WHT	1	TD +
TD - 2	OR	GRN	2	TD -
RD + 3	GRN/WHT	OR/WHT	3	RD +
4	BLU	BLU	4	
5	BLU/WHT	BLU/WHT	5	
RD - 6	GRN	OR	6	RD -
7	BRN/WHT	BRN/WHT	7	
8	BRN	BRN	8	

RJ45 RJ45

8-pin RJ45

Communication Settings

Communications Port

This parameter is used to select the port number to be used by the driver for communicating with the remote device. The default value is 28784 (0x7070) for non-GS device models (H2-EBC, H2-EBC-F, H4-EBC, H4-EBC-F, and Terminator I/O). For GS device models (GS1, GS2 and GS3) the default value is 502. Legal port number values are between 0-

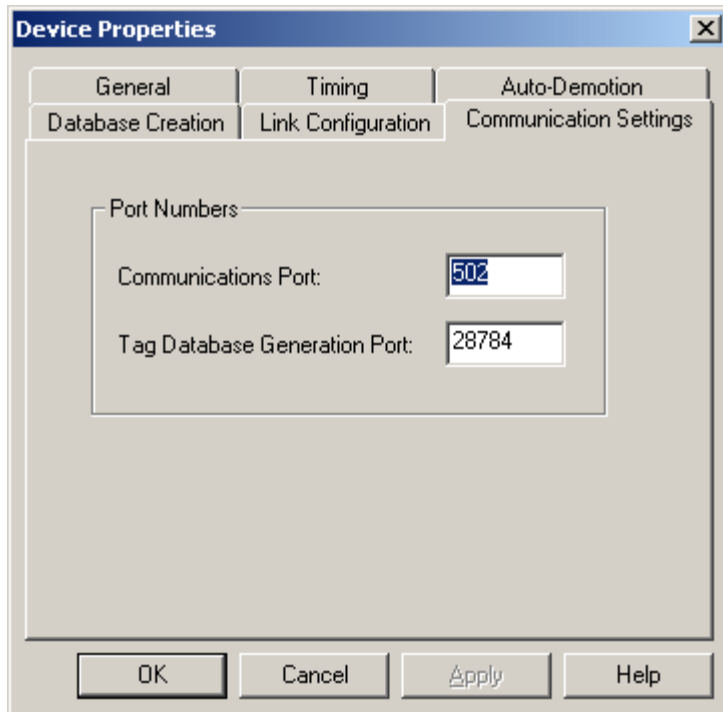
65535.

Tag Database Generation Port

This parameter is used to select the port number to be used by the driver for performing auto tag database generation from the remote device. This option is available only for GS device models (GS1, GS2 and GS3). The default value is 28784 (0x7070). Legal port number values are between 0-65535.

Note 1: For non-GS device models, the Communications Port value is also used for Tag Database Generation. The Tag Database Generation Port option is not visible when a non-GS device model is selected.

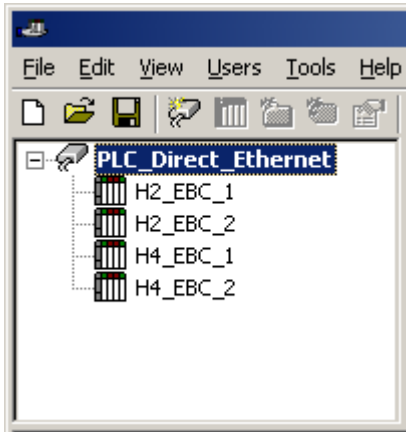
Note 2: The Communication Settings can be reached when defining a device in the Device Wizard and afterward by clicking **Device Properties | Communication Settings**.



Optimizing Your AutomationDirect EBC Communications

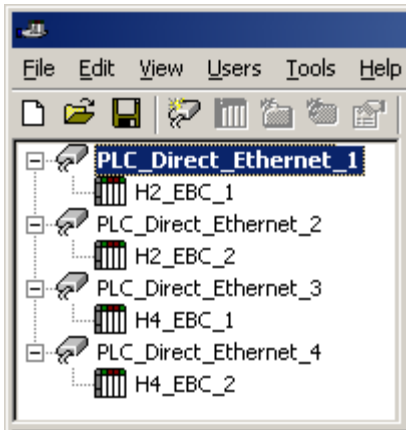
The AutomationDirect EBC driver has been designed to provide the best performance with the least amount of impact on the system's overall performance. While the AutomationDirect EBC driver is fast, there are a couple of guidelines that can be used in order to control and optimize the application and gain maximum performance.

Our server refers to communications protocols like AutomationDirect EBC as a channel. Each channel defined in the application represents a separate path of execution in the server. Once a channel has been defined, a series of devices must then be defined under that channel. Each of these devices represents a single AutomationDirect controller from which data will be collected. While this approach to defining the application will provide a high level of performance, it won't take full advantage of the AutomationDirect EBC driver or the network. An example of how the application may appear when configured using a single channel is shown below.



Each device appears under a single AutomationDirect EBC channel. In this configuration, the driver must move from one device to the next as quickly as possible in order to gather information at an effective rate. As more devices are added or more information is requested from a single device, the overall update rate begins to suffer.

If the AutomationDirect EBC driver could only define one single channel, then the example shown above would be the only option available; however, the AutomationDirect EBC driver can define up to 100 channels. Using multiple channels distributes the data collection workload by simultaneously issuing multiple requests to the network. An example of how the same application may appear when configured using multiple channels to improve performance is shown below.



Each device has now been defined under its own channel. In this new configuration, a single path of execution is dedicated to the task of gathering data from each device. If the application has 100 or fewer devices, it can be optimized exactly how it is shown here.

The performance will improve even if the application has more than 100 devices. While 100 or fewer devices may be ideal, the application will still benefit from additional channels. Although spreading the device load across all channels will cause the server to move from device to device again, it can now do so with far less devices to process on a single channel.

Data Types Description

Data Type	Description
Boolean	Single bit
Byte	Unsigned 8 bit value bit 0 is the low bit bit 7 is the high bit
Char	Signed 8 bit value bit 0 is the low bit bit 6 is the high bit bit 7 is the sign bit
Word	Unsigned 16 bit value bit 0 is the low bit bit 15 is the high bit
Short	Signed 16 bit value bit 0 is the low bit bit 14 is the high bit

	bit 15 is the sign bit
DWord	Unsigned 32 bit value bit 0 is the low bit bit 31 is the high bit
Long	Signed 32 bit value bit 0 is the low bit bit 30 is the high bit bit 31 is the sign bit
Float	32 bit floating point value (EBC Only) 16 bit floating point value (GS-EDRV Only)

Address Descriptions

Address specifications vary depending on the model in use. Select a link from the following list to obtain specific address information for the model of interest.

H2-EBC(-F)

[I/O Addressing](#)

[Serial Port Addressing](#)

H4-EBC(-F)

[I/O Addressing](#)

[Serial Port Addressing](#)

Terminator I/O

[I/O Addressing](#)

[Serial Port Addressing](#)

GS-EDRV

[GS Drive Parameter Addressing](#)

[GS Drive Status Addressing](#)

H2-EBC(-F) Addressing

There are two types of addressing for H2-EBC(-F) models, I/O and serial port. Select a link from the list below for more information on a specific address type.

[I/O Addressing](#)

[Serial Port Addressing](#)

H4-EBC(-F) Addressing

There are two types of addressing for H4-EBC(-F) models, I/O and serial port. Select a link from the list below for more information on a specific address type.

[I/O Addressing](#)

[Serial Port Addressing](#)

Terminator I/O Addressing

There are two types of addressing for Terminator I/O models, I/O and serial port. Select a link from the list below for more information on a specific address type.

[I/O Addressing](#)

[Serial Port Addressing](#)

Status Addressing

Extended Error information is available for referencing in certain modules. The Extended Error contains module specific error information that a single Error Code could not define on its own. This data is available regardless if the module is in an error state.

Currently only the T1F-14THM module supports addressing of Extended Error data. Each Byte represents a Thermocouple Channel. The general form this Status information is as follows: S<ss>:STS.EXT<nn> where ss is the slot number (1 to 93), and nn is the 0-based channel number (0 - 63).

Example

S1:STS.EXT0 = Extended Error for Thermocouple Channel 1.

See Also: [Application Notes: T1F-14THM](#)

GS Drive Addressing

There are two types of addressing for GS-EDRV models, parameter and status variables. Select a link from the list below for more information on a specific address type.

[GS Drive Parameter Addressing](#)

[GS Drive Status Addressing](#)

H2-EBC(-F) I/O Addressing

Data Types

The EBC is designed to replace the PLC CPU. As a result, I/O slots must be individually addressed as there is no memory addressing. The general form for H2 addresses is as follows: S<ss>:<t><nn> where ss is the slot number (0 to 14), t is the address type (DI, DO, WI, WO), and nn is the address. The address ranges from 0 to an upper limit determined by the module occupying the slot.

Note: The default data types are shown in **bold**.

Address Type	Range	Data Type
Discrete Inputs (X: Point)	X<nn>, DI<nn> nn = Bit Number (decimal)	Boolean , Byte, Char, Word, Short, DWord, Long
Discrete Outputs (Y: Point)	Y<nn>, DO<nn> nn = Bit Number (decimal)	Boolean , Byte, Char, Word, Short, DWord, Long
Byte Inputs	BI<nn> nn = Byte Number (decimal)	Byte , Char
Byte Outputs	BO<nn> nn = Byte Number (decimal)	Byte , Char
Word Inputs (K: Analog Channel)	K<nn>, WI<nn> nn = Word Number (decimal)	Word , Short
Word Outputs (V: Analog Channel)	V<nn>, WO<nn> nn = Word Number (decimal)	Word , Short
DWord Inputs	DWI<nn> nn = DWord Number (decimal)	DWord , Long
DWord Outputs	DWO<nn> nn = DWord Number (decimal)	DWord , Long
Float Inputs	FI<nn> nn = Float Number (decimal)	Float
Float Outputs	FO<nn> nn = Float Number (decimal)	Float
Double Inputs	DBI<nn> nn = Double Number (decimal)	Float
Double Outputs	DBO<nn> nn = Double Number (decimal)	Float

Example

The following example illustrates a sample arrangement and its corresponding addresses.

H2-EBC Module	Slot 0 8 Inputs	Slot 1 32 Inputs	Slot 2 4 Analog In	Slot 3 8 Outputs	Slot 4 16 Outputs	Slot 5 8 Analog Out
	Addresses S0:X0	Addresses S1:X0	Addresses S2:K0	Addresses S3:Y0	Addresses S4:Y0	Addresses S5:V0
	↓	↓	↓	↓	↓	↓
	∨ S0:X7	∨ S1:X31	∨ S2:K3	∨ S3:Y7	∨ S4:Y15	∨ S5:V7

Case: Various valid and invalid item entries and their meaning.

S0:X0	Bit 0 of slot 0 input
S3:K4	Word 4 of slot 3 input

Case: 32-point output module in slot 1.

S1:Y0@Byte	Byte 0 of slot 1 output
S1:Y3@Byte	Invalid Address Must be on even byte boundary
S1:Y8@Byte	Byte 1 of slot 1 output
S1:Y16@Word	Word 1 of slot 1 output

Case: 4-channel in/ 2-channel out module in slot 2.

S2:K3	Word 3 of slot 2 input
S2:V1	Word 1 of slot 2 output

H4-EBC(-F) I/O Addressing**Data Types**

The EBC is designed to replace the PLC CPU. As a result, I/O bases and slots must be individually addressed as there is no memory addressing. The general form for H4 addresses is as follows: *B<bb>:S<ss>:<t><nn>* where *bb* is the base number (0 to 3), *ss* is the slot number (0 to 14), *t* is the address type (DI, DO, WI, WO), and *nn* is the address. The address ranges from 0 to an upper limit determined by the module occupying the slot.

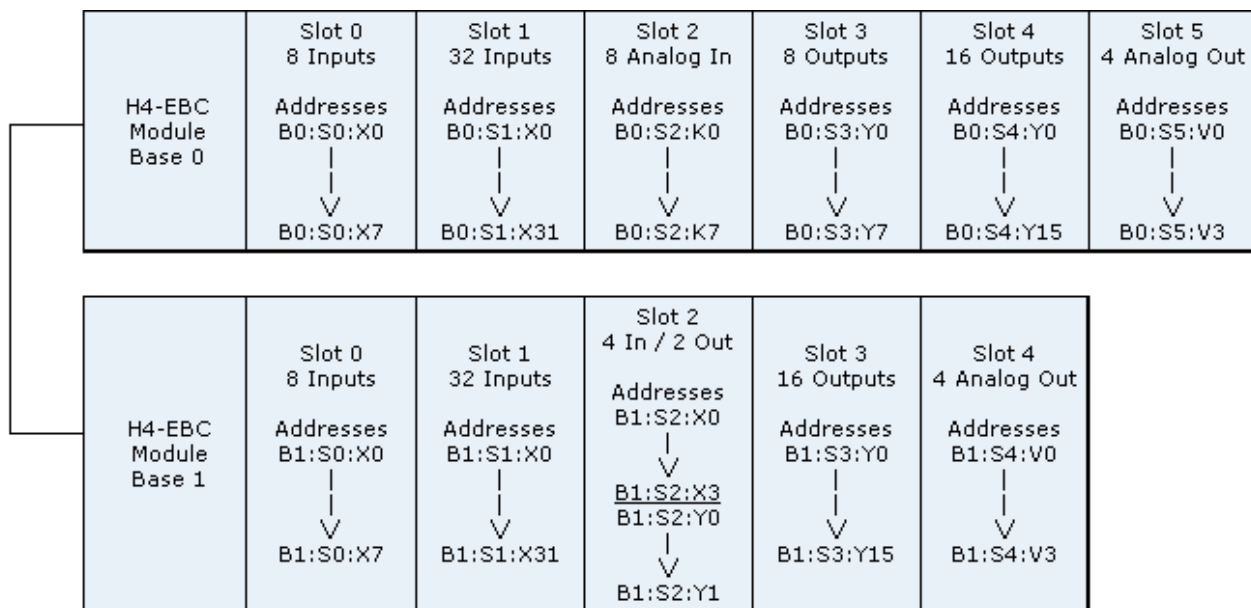
Note: The default data types are shown in **bold**.

Address Type	Range	Data Type
Discrete Inputs (X: Point)	X<nn>, DI<nn> nn = Bit Number (decimal)	Boolean , Byte, Char, Word, Short, DWord, Long
Discrete Outputs (Y: Point)	Y<nn>, DO<nn> nn = Bit Number (decimal)	Boolean , Byte, Char, Word, Short, DWord, Long
Byte Inputs	BI<nn> nn = Byte Number (decimal)	Byte , Char
Byte Outputs	BO<nn> nn = Byte Number (decimal)	Byte , Char
Word Inputs (K: Analog Channel)	K<nn>, WI<nn> nn = Word Number (decimal)	Word , Short
Word Outputs (V: Analog Channel)	V<nn>, WO<nn> nn = Word Number (decimal)	Word , Short
DWord Inputs	DWI<nn> nn = DWord Number (decimal)	DWord , Long
DWord Outputs	DWO<nn> nn = DWord Number (decimal)	DWord , Long

Float Inputs	FI<nn> nn = Float Number (decimal)	Float
Float Outputs	FO<nn> nn = Float Number (decimal)	Float
Double Inputs	DBI<nn> nn = Double Number (decimal)	Float
Double Outputs	DBO<nn> nn = Double Number (decimal)	Float

Example

The following example illustrates a sample arrangement and its corresponding addresses.



Case: Various valid and invalid item entries and their meaning.

B0:S0:X0	Bit 0 of slot 0 input, base 0
S3:K4	Error, need to specify base

Case: 32-point output module in slot 1, base 3.

B3:S1:Y0@Byte	Byte 0 of slot 1 output
B3:S1:Y3@Byte	Invalid Address Must be on even byte boundary
B3:S1:Y8@Byte	Byte 1 of slot 1 output
B3:S1:Y16@Word	Word 1 of slot 1 output

Case: 4-channel in/ 2-channel out module in slot 2, base 0.

B0:S2:K3	Word 3 of slot 2 input
B0:S2:V1	Word 1 of slot 2 output

Terminator I/O I/O Addressing

Data Types

The EBC is designed to replace the PLC CPU. As a result, I/O slots must be individually addressed as there is no memory addressing. The general form for Terminator I/O addresses is as follows: S<ss>:<t><nn> where ss is the slot number (1 to 93), t is the address type (DI, DO, WI, WO, etc.), and nn is the address. The address ranges from 0 to an upper limit determined by the module occupying the slot.

Note: The default data types are shown in **bold**.

Address Type	Range	Data Type
Discrete Inputs	DI<nn> nn = Bit Number (decimal)	Boolean , Byte, Char, Word, Short, DWord, Long
Discrete Outputs	DO<nn> nn = Bit Number (decimal)	Boolean , Byte, Char, Word, Short, DWord, Long
Byte Inputs	BI<nn> nn = Byte Number (decimal)	Byte , Char
Byte Outputs	BO<nn> nn = Byte Number (decimal)	Byte , Char
Word Inputs	WI<nn> nn = Word Number (decimal)	Word , Short
Word Outputs	WO<nn> nn = Word Number (decimal)	Word , Short
DWord Inputs	DWI<nn> nn = DWord Number (decimal)	DWord , Long
DWord Outputs	DWO<nn> nn = DWord Number (decimal)	DWord , Long
Float Inputs	FI<nn> nn = Float Number (decimal)	Float
Float Outputs	FO<nn> nn = Float Number (decimal)	Float
Double Inputs	DBI<nn> nn = Double Number (decimal)	Float
Double Outputs	DBO<nn> nn = Double Number (decimal)	Float

Example

The following example illustrates a sample arrangement and its corresponding addresses.

Terminator IO EBC Module	Slot 1 8 Digital In	Slot 2 16 Digital In	Slot 3 8 Digital Out	Slot 4 16 Digital Out	Slot 5 8 Analog In	Slot 6 16 Analog Out
Slot 0 Serial IO Ports	Addresses S1:DI0	Addresses S2:DI0	Addresses S3:DO0	Addresses S4:DO0	Addresses S5:DWI0	Addresses S6:DWO0
EBC:SP0.Item	↓ S1:DI7	↓ S2:DI15	↓ S3:DO7	↓ S4:DO15	↓ S5:DWI7	↓ S6:DWO15

Case: Various valid and invalid item entries and their meaning.

S0:DI0	Discrete inputt 0 of slot 0 input
S3:WI4	Word 4 of slot 3 input

Case: 32-point output module in slot 1.

S1:DO0@Byte	Byte 0 of slot 1 output
S1:DO3@Byte	Invalid Address Must be on even byte boundary
S1:DO8@Byte	Byte 1 of slot 1 output
S1:DO16@Word	Word 1 of slot 1 output

Case: 4-channel in/ 2-channel out module in slot 2.

S2:WI3	Word 3 of slot 2 input
--------	------------------------

S2:W01 Word 1of slot 2 output

H2, H4, Terminator I/O Serial Port Addressing

The EBC contains an RS232 serial port. Both the transmit buffer and receive buffer of the driver are 127 bytes in size. Likewise, the corresponding tags can be a maximum of 127 bytes. Incoming bytes are appended to the receive buffer. If the receive buffer is full or additional bytes will overflow the buffer, the buffer will reset with these additional bytes. The H2-SERIO module contains 3 serial ports each identical to the EBC serial port. The EBC serial port and the H2-SERIO serial ports differ only in the way the ports are addressed.

The following is a list of possible configurations.

1. Baud rates: 9600
2. Data bits: 7 or 8
3. Parity: none, odd, or even
4. Stop bits: 1 or 2

Note 1: The device will not maintain configuration values while it's power is off. Upon power-up, the values will need to be re-entered.

Note 2: An RJ45 connector is required.

Specifying A Port

Port specifiers precede the serial port address. It defines which port the serial port address corresponds to. For addressing the EBC serial port, no base or slot information is needed. To define an EBC address the mnemonic "EBC" is used. The mnemonic SP0 means serial port 0. The EBC serial port is always serial port 0.

For addressing serial ports in modules, a base (if applicable) and slot must be specified. This is similar to I/O addressing. $\langle bb \rangle$ = base number in which the module is located, $\langle ss \rangle$ = slot number in which the module is located, and $\langle pp \rangle$ specifies the port number of interest. Starting in slot 0 (or 1 for Terminator I/O), the first serial port module encountered is assigned ports 1 - (1 + # module 1 serial ports). Let $x = (1 + \# \text{ module 1 serial ports})$. The second serial port module encountered is assigned ports $(x - (x + \# \text{ module 2 serial ports}))$.

The table below summarizes port specification for both EBC and serial port module addressing.

Location	H2/Terminator I/O Syntax	H4 Syntax
EBC (Base)	EBC:SP0	EBC:SP0
Module	S $\langle ss \rangle$:SP $\langle pp \rangle$	B $\langle bb \rangle$:S $\langle ss \rangle$:SP $\langle pp \rangle$

Example

EBC = H2

Slot 1 = 3 port serial port module

Slot 5 = 3 port serial port module

To specify the 3rd port in Slot 1, enter:

S1:SP3

To specify the 3rd port in Slot 5, enter:

S5:SP6

Specifying an Address

For the following topics, $\langle port\ spec \rangle$ will be a placeholder for the port specifier previously discussed. Examples are given to illustrate the combination of port specifier + serial port address and example values they may undertake.

[Serial Data Received](#)

[Number of RX Queue Bytes](#)

[Flush Data Received](#)

[Flush RX Queue](#)

[Serial Data To Transmit](#)

[Number of TX Queue Bytes](#)[Flush TX Queue](#)[Serial Port Configuration: Baud Rate](#)[Serial Port Configuration: #Data Bits](#)[Serial Port Configuration: Parity](#)[Serial Port Configuration: #Stop Bits](#)[Serial Port Configuration: Mode](#)[Serial Port Configuration: Use RTS Line](#)[Serial Port Configuration: Pre-Transmit Delay](#)[Serial Port Configuration: Post-Transmit Delay](#)

Serial Data Received

To read serial data received, reference address type `<port spec>.DATAIN`. All bytes read will be removed from the serial port's RX queue upon reading `DATAIN`.

Requirements

Serial port configuration addresses must have appropriate values for proper operation.

Note: The default data types are shown in **bold**.

Syntax	Data Type	Number of Bytes Read	Access
<code><port spec>.DATAIN [r][c]*</code>	Byte Array , Char Array	r times c	Read Only
<code><port spec>.DATAIN</code>	String	127	Read Only

*To access as an array, `[row][column]` form is required. For example, `DATAIN [1][24]` would display 24 ASCII bytes in array notation: `[x1, x2, x3..x24]`

Examples

Address	Value	Description
<code>EBC.SP0.DATAIN</code>	"hello"	port 0 input data viewed as a string.
<code>S3.SP2.DATAIN [2][2]</code>	<code>[105, 105][105, 105]</code>	port 2 input data in array form. In string form this would equate to "iiii".

Number of RX Queue Bytes

The number of bytes in the serial port's RX queue can be accessed by referencing address type `RXAVAIL`.

Requirements

None.

Note: The default data types are shown in **bold**.

Syntax	Data Type	Access
<code><port spec>.RXAVAIL</code>	Word , Short	Read Only

Examples

Address	Value	Description
<code>EBC.SP0.RXAVAIL</code>	0	port 0, no bytes in RX queue.
<code>B0.S1:SP1.RXAVAIL</code>	5	port 1, 5 bytes in RX queue.

Flush Data Received

To flush the local `DATAIN` buffer, reference `DIFLUSH`.

Values

true = flush DATAIN
false = do nothing

Requirements

Serial port configuration addresses must have appropriate values for proper operation.

Note: The default data types are shown in **bold**.

Syntax	Data Type	Access
<port spec>.DIFLUSH	Boolean	Write Only

Examples

Address	Value	Description
EBC:SP0.DIFLUSH	true	flush port 0's DATAIN buffer.

Flush RX Queue

To flush the serial port's RX queue, reference *RXFLUSH*. All bytes in the serial port's RX queue will be lost and therefore not accessible on sub sequential reads of *DATAIN*.

Values

true = flush queue
false = do nothing

Requirements

Serial port configuration addresses must have appropriate values for proper operation.

Note: The default data types are shown in **bold**.

Syntax	Data Type	Access
<port spec>.RXFLUSH	Boolean	Write Only

Examples

Address	Value	Description
EBC:SP0.RXFLUSH	true	flush port 0's RX queue.

Serial Data To Transmit

To transmit serial data, reference address type *DATAOUT*. All bytes written will be removed from the serial port's TX queue upon completion of transmission.

Requirements

Serial port configuration addresses must have appropriate values for proper operation.

Note: The default data types are shown in **bold**.

Syntax	Data Type	Number of Bytes Read	Access
<port spec>.DATAOUT [r][c]*	Byte Array , Char Array	r times c	Write Only
<port spec>.DATAOUT	String	length of null terminated string	Write Only

*To access as an array, *[row][column]* form is required. For example, *DATAOUT [1][24]* would send 24 ASCII bytes in array notation: *[x1, x2, x3..x24]*)

Examples

Address	Value	Description
EBC.SP0.DATAOUT	"hello"	port 0, transmit "hello".

S3.SP2.DATAOUT [2][2]	[105, 105][105, 105]	port 2 data to transmit in array form. In string form this would equate to "iiii".
-----------------------	----------------------	--

Number of TX Queue Bytes

Reference *TXLEFT* to determine how many bytes are left in the TX queue. This value is available after transmission of serial data.

Requirements

None.

Note: The default data types are shown in **bold**.

Syntax	Data Type	Access
<port spec>.TXLEFT	Word , Short	Read Only

Examples

Address	Value	Description
EBC.SP0.TXLEFT	0	port 0, no bytes left in queue, all bytes on last transmission have been sent
B0.S1:SP1.TXLEFT	10	port 1, 10 bytes have yet to be sent and reside in the TX queue.

Flush TX Queue

To flush the serial port's TX queue, reference *TXFLUSH*. All bytes in the serial port's TX queue will be lost and therefore not sent to the target serial device. *TXLEFT* will reset upon flushing of the TX queue.

Values

true = flush queue
false = do nothing

Requirements

None.

Note: The default data types are shown in **bold**.

Syntax	Data Type	Access
<port spec>.TXFLUSH	Boolean	Write Only

Examples

Address	Value	Description
EBC.SP0:TXFLUSH	true	flush port 0's TX queue.

Serial Port Configuration: Baud Rate

To configure the baud rate for a serial port, reference *BAUD*.

Values

9600

Requirements

None.

Note: The default data types are shown in **bold**.

Syntax	Data Type	Access
--------	-----------	--------

<port spec>.BAUD	DWord , Long	Read/Write
------------------	---------------------	------------

Examples

Address	Value	Description
EBC.SP0.BAUD	9600	port 0, baud = 9600.

Serial Port Configuration: #Data Bits

To configure the number of data bits for a serial port, reference *DATABITS*.

Values

7 or 8

Requirements

None.

Note: The default data types are shown in **bold**.

Syntax	Data Type	Access
<port spec>.DATABITS	Byte , Char	Read/Write

Examples

Address	Value	Description
EBC.SP0.DATABITS	7	port 0 configured for 7 databits.

Serial Port Configuration: Parity

To configure the parity for a serial port, reference *PARITY*.

Values

0, 1 = none
2 = odd
3 = even

Requirements

None.

Note: The default data types are shown in **bold**.

Syntax	Data Type	Access
<port spec>.PARITY	Byte , Char	Read/Write

Examples

Address	Value	Description
EBC.SP0.PARITY	0	port 0 configured for no parity.

Serial Port Configuration: #Stop Bits

To configure the number of stop bits for a serial port, reference *STOPBITS*.

Values

1 or 2

Requirements

None.

Note: The default data types are shown in **bold**.

Syntax	Data Type	Access
<port spec>.STOPBITS	Byte , Char	Read/Write

Examples

Address	Value	Description
EBC.SP0.STOPBITS	1	port 0 configured for 1 databits.

Serial Port Configuration: Mode

To configure the mode of operation for the serial port, reference *MODE*.

Values

0 = KSequence Slave (not supported)
1 = Proxy (ASCII communications)

Requirements

None.

Note: The default data types are shown in **bold**.

Syntax	Data Type	Access
<port spec>.MODE	Byte , Char	Read/Write

Examples

Address	Value	Description
EBC.SP0.MODE	1	port 0 configured for generic ASCII communications.

Serial Port Configuration: Use RTS Line

Flow control may be required to communicate with certain serial devices. A flow control option available is *USERTS*. This specifies whether the RTS line is to be used or not. The RTS line will be high if bytes are available for transmission. After all buffered bytes have been sent; the RTS line will be low. This is normally used with RS232/RS485 converter hardware.

Values

true = Use the RTS line
false = Don't use the RTS line

Requirements

None.

Note: The default data types are shown in **bold**.

Syntax	Data Type	Access
<port spec>.USERTS	Boolean	Read/Write

Examples

Address	Value	Description
EBC.SP0.USERTS	true	port 0 configured to use the RTS line.

Serial Port Configuration: Pre-Transmit Delay

Reference *PRETXDLY* to configure how long the RTS line is to be high before beginning transmission.

Values

Delay = PRETXDLY value milliseconds times 2.

Requirements

USERTS = true.

Note: The default data types are shown in **bold**.

Syntax	Data Type	Access
<port spec>.PRETXDLY	Byte , Char	Read/Write

Examples

Address	Value	Description
EBC.SP0.PRETXDLY	2	port 0 configured to delay 4 ms.

Serial Port Configuration: Post-Transmit Delay

Reference *POSTTXDLY* to configure how long the RTS line is to be held high after transmission.

Values

Delay = POSTTXDLY value milliseconds times 2.

Requirements

USERTS = true.

Note: The default data types are shown in **bold**.

Syntax	Data Type	Access
<port spec>.POSTTXDLY	Byte , Char	Read/Write

Examples

Address	Value	Description
EBC.SP0.POSTTXDLY	1	port 0 configured to delay 2 ms.

GS Drive Parameter Addressing

GS Drives are addressed by means of parameters and parameter groups. Both are specific to the drive in question and can be found by consulting the drive's user manual

Addressing Syntax

Both group and parameter are 0-based. All parameters are read/write.

P<group>.<parameter>

Parameter Definitions

Definitions for both parameter groups and parameters can be found in the particular drive's user manual.

Drive	Manual
GS1	GS1-M
GS2	GS2-M
GS3	GS3-M

Float Data

Parameters with default data type Float, also support Word and Short access. Float data is returned from the drives in raw form (no decimal point). Referencing a float parameter as Word and Short will return this raw value while referencing it as a Float will return the scaled value in floating point notation.

Examples

1. Drive: GS1

- Parameter Group: Ramps
- Parameter: Acceleration Time 1
- Address: **P1.1**
- Data Type: Float, Value = 10.0
- Data Type: Word, Value = 100

2. Drive: GS2

- Parameter Group: Analog
- Parameter: Analog Input Reverse Motion Enable
- Address: **P4.4**
- Data Type: Word

GS Drive Status Addressing

GS Drives also contain status information much the same way parameters are addressed. Available status variables are specific to the drive in question and can be found by consulting the drive's user manual

Addressing Syntax

Both group and status variable are 0-based. Some status variables are read only.

ST<group>.<status variable>

For now, only group 0 is available.

Status Definitions

Group	Status Variable	Model	Label	Data Type
0	0	GS1/GS2/GS3	Status_Monitor_1	Word
0	1	GS1/GS2/GS3	Status_Monitor_2	Word
0	2	GS1/GS2/GS3	Frequency_Command_F	Float
0	3	GS1/GS2/GS3	Output_Frequency_H	Float
0	4	GS1/GS2/GS3	Output_Current_A	Float
0	5	GS1/GS2/GS3	DC_BUS Voltage_U	Float
0	6	GS1/GS2/GS3	Output_Voltage_E	Float
0	7	GS1/GS3	Motor_RPM	Word
0	8	GS1/GS3	Scale_Frequency_Low	Word
0	9	GS1/GS3	Scale_Frequency_High	Word
0	10	GS2/GS3	Power_Factor_Angle	Word
0	11	GS1/GS3	Percent_Load	Word
0	12	GS3	PID_Setpoint	Word
0	13	GS3	PID_Feedback_Signal	Word
0	14	N/A	Reserved	N/A
0	15	N/A	Reserved	N/A
0	16	GS1/GS2/GS3	Software_Version	Word
0	17	N/A	Reserved	N/A
0	18	GS1/GS2/GS3	Serial_Status	Word

Note: Definitions for status variables can be found in the specific driver's user manual.

Drive	Manual
GS1	GS1-M
GS2	GS2-M
GS3	GS3-M

Float Data

Status variables with default data type Float, also support Word and Short access. Float data is returned from the drives in raw form (no decimal point). Referencing a float status variable as Word and Short will return this raw value while referencing it as a Float will return the scaled value in floating point notation. See examples below.

Examples

1. Drive: GS1
 - Status: Status Monitor 1
 - Address: **ST0.0**
 - Data Type: Word

2. Drive: GS3
 - Status Variable: DC BUS Voltage U
 - Address: **ST0.5**
 - Data Type: Float, Value = 24.0
 - Data Type: Word, Value = 240

Module Hot Swapping

Terminator I/O allows for the Hot Swapping of I/O modules (blue component) while the power is on. AutomationDirect.com states the following I/O module replacement procedure:

1. Remove the I/O module from base.
2. Install the new I/O module of the same part number.
3. Verify that the Base Controller LEDs have returned to normal.

Note 1: I/O bases (black component) are not hot swappable.

Note 2: It is strongly recommended that modules are hot swapped one at a time.

What If I Don't Follow The Preferred Replacement Procedure?

A replacement procedure differing from the above may produce erroneous results from the server. It is very important that the new module is the same size as the previous module. Size refers to the number of discretes/bytes/words/DWords/floats/etc. If the previous module had 8 DWords and 8 Discretes, the new module must have 8 DWords and 8 Discretes. The safest measure to take is to replace a module with a module of the same part number.

If the sizes do not match, communication errors will occur and OPC data will go Bad. To remedy this problem, cycle the device's power when safe and appropriate. It is possible to remain in an erroneous state after cycling the device's power. This condition may occur if the replacement module has different address types than the original module. In this case, client tags for this module are referencing invalid addresses. The server project and possibly the client project will need editing to correct this problem. Again, this problem will not occur if the replacement module and original module have the same part number.

Error Descriptions

Driver Error Messages

[Winsock initialization failed \(OS Error = n\)](#)

[Winsock V1.1 or higher must be installed to use the AutomationDirect EBC device driver](#)

[Memory allocation error](#)

Driver Warning Messages

[Device '<device name>' not responding](#)

[Unable to write to '<address>' on device '<device name>'](#)

[Device address '<address>' contains a syntax error](#)

[Address '<address>' is out of range for the specified device or register](#)

[Device address '<address>' is not supported by model '<model name>'](#)

[Device address '<address>' is Read Only](#)

[Data type '<type>' is not valid for device address '<address>'](#)

[Device '<device name>' returned an error with value = '<error value>' disabling Link Watchdog. All tags will be invalidated](#)

[Device '<device name>' returned an error with value = '<error value>' enabling Link Watchdog with timeout = '<timeout>'. All tags will be invalidated](#)

Read Errors

[Address '<address>' is out of range for device '<device name>'. Tag Deactivated](#)

[Base referenced in address '<address>' on device '<device name>' does not exist. Tag Deactivated](#)

[Module referenced in address '<address>' on device '<device name>' does not exist. Tag Deactivated](#)

[Invalid address type for address '<address>' on device '<device name>'. Tag Deactivated](#)

[Device '<device name>' returned an error code with value = '<value>' reading address '<address>'](#)

[Device '<device name>' : Base '<base>' : Slot '<slot>' returned an error with value = '<error value>'](#)

[Device '<device name>' : Base '<base>' : Slot '<slot>' returned a warning with value = '<warning value>'](#)

[Device '<device name>' : Base '<base>' : Slot '<slot>' returned information with value = '<info value>'](#)

[Device '<device name>' : Base '<base>' : Slot '<slot>' returned an internal error with value = '<internal value>'](#)

[Device '<device name>' : Base '<base>' : Slot '<slot>' returned extended error: Type = '<ext err type>', Data = '<error bytes>'](#)

[Frame received from device '<device name>' contains errors](#)

[Frame received from device '<device name>' contains errors. Verify its IP address](#)

[Format version of frame received from device '<device name>' is not supported. Tag Deactivated](#)

[Model selected for device '<device name>' does not match the actual device model. Verify its IP address](#)

[Device '<device name>' block request \[<start> to <end>\] responded with exception <code>](#)

[Bad received length \[<start> to <end>\] on device '<device name>'](#)

Write Errors

[Write rejected. Address '<address>' is out of range for device '<device name>'](#)

[Write rejected. Base referenced in address '<address>' on device '<device name>' does not exist](#)

[Write rejected. Module referenced in address '<address>' on device '<device name>' does not exist](#)

[Write rejected. Invalid address type for address '<address>' on device '<device name>'](#)

[Write to address '<address>' failed. Device '<device name>' returned an undefined status code with value = '<value>'](#)

[Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned an error with value = '<error value>'](#)

[Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned a warning with value = '<warning value>'](#)

[Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned information with value = '<info value>'](#)

[Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned an internal error with value = '<internal value>'](#)

[Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned extended error: Type = '<ext err type>', Data = '<error bytes>'](#)

[Write to address '<address>' failed. Frame received from device '<device name>' contains errors](#)

[Write to address '<address>' failed. Frame received from device '<device name>' contains errors. Verify its IP address](#)

[Write to address '<address>' failed. Format version of frame received from device '<device name>' is not supported](#)

[Write to address '<address>' failed. Model selected for device '<device name>' does not match the actual device model. Verify its IP address](#)

[Unable to write to address <address>. Device '<device name>' responded with exception <code>](#)

EBC Errors/Warnings/Information Return Values & Description

Value	Description
111	Invalid type.
112	RAM already locked.
113	Invalid request.
114	Timeout error.
115	Flash program error.
116	Invalid OS.
117	Invalid location (ie. Write attempted to an invalid analog channel).
118	Invalid slot number.
119	Invalid data.
120	Module busy.
121	Analog Input Channel failure; nn contains channel number that failed.
122	Unused analog input channels exist.
123	Invalid UDP port.
124	Shutdown OS.
125	Invalid IP address.
126	Protection error.
127	Unknown type.
128	Backplane initialization error.
129	Unknown response.
130	Unknown read/write format.
131	Unknown ACK.
132	Unknown NAK.
133	Range error.
134	Length warning.
135	Invalid base number.
136	Invalid module type.
137	Invalid offset.
138	Invalid boot version for OS.
139	Broken transmitter; nn contains channel number that failed.
140	Invalid address.
142	Channel fail multiple; nn contains channel BITS from module. Example: If bit 0 is set then channel 0 has failed If bit 1 and 3 are set then channels 1 and 3 have failed.
153	I/O module missing (I/O module removed in hot swap).
154	I/O Base has changed (I/O module replaced in hot swap).
155	Module in error. Possible errors: - missing 24V on discrete modules - blown fuses on discrete modules - missing 24V on analog modules - missing CJC block on the T1F-14THM
200-216	XX unused analog input channels exist where: XX = Value - 200.

> 32 (0x20)	BIT Type of Error
and	0 Terminal block off
< 64 (0x40)	1 External P/S voltage low
for 405 Family	2 Fuse blown
	3 Bus Error
	4 Module init error (intelligent module)
	5 Faults exist in module (this bit is set if any of the above bits are set)
	Example: 0x22: External P/S Voltage low
0x8000	Function not implemented.
0x8001	Version passed to function not correct for library.
0x8002	Supplied transport not supported.
0x8003	Supplied device is not valid.
0x8004	Supplied buffer is too small.
0x8005	Zero bytes were returned in the packet.
0x8006	Timeout error.
0x8007	Supplied protocol not supported.
0x8008	The device's IP address has not been set.
0x8009	No transport specified.
0x800A	IPX transport not installed.
0x800B	Error opening IPX Socket.
0x800C	No packet driver found.
0x800D	CRC did not match.
0x800E	Memory allocation error failed.
0x800F	No cache has been allocated for IPX.
0x8010	Invalid request.
0x8011	No response was available.
0x8012	Invalid format response was received.
0x8013	Given data is too large.
0x8014	Error loading procedures.
0x8015	Attempted command before successful OpenTransport.
0x8016	Data not aligned on proper boundary.

Driver Error Messages

The following error/warning messages may be generated. Click on the link for a description of the message.

[Winsock initialization failed \(OS Error = n\)](#)

[Winsock V1.1 or higher must be installed to use the AutomationDirect EBC device driver](#)

[Memory allocation error](#)

Winsock initialization failed (OS Error = n)

Error Type:

Fatal

OS Error	Indication	Possible Solution
10091	Indicates that the underlying network subsystem is not ready for network communication.	Wait a few seconds and restart the driver.
10067	Limit on the number of tasks supported by the Windows Sockets implementation has been reached.	Close one or more applications that may be using Winsock and restart the driver.

Winsock V1.1 or higher must be installed to use the AutomationDirect EBC device driver

Error Type:

Fatal

Possible Cause:

The version number of the Winsock DLL found on the system is less than 1.1.

Solution:

Upgrade Winsock to version 1.1 or higher.

Memory allocation error

Error Type:

Fatal

Possible Cause:

Insufficient system RAM to support the number of tags the driver is being asked to scan.

Solution:

Increase the amount of system memory or reduce the number tags being scanned.

Driver Warning Messages

The following error/warning messages may be generated. Click on the link for a description of the message.

[Device '<device name>' not responding](#)

[Unable to write to '<address>' on device '<device name>'](#)

[Device address '<address>' contains a syntax error](#)

[Address '<address>' is out of range for the specified device or register](#)

[Device address '<address>' is not supported by model '<model name>'](#)

[Device address '<address>' is Read Only](#)

[Data type '<type>' is not valid for device address '<address>'](#)

[Device '<device name>' returned an error with value = '<error value>' disabling Link Watchdog. All tags will be invalidated](#)

[Device '<device name>' returned an error with value = '<error value>' enabling Link Watchdog with timeout = '<timeout>'. All tags will be invalidated](#)

Device '<device name>' not responding

Error Type:

Serious

Possible Cause:

1. The Ethernet connection between the device and the host PC is broken.
2. The named device may have been assigned an incorrect IP address.
3. The requested address is not available in the device.

Solution:

1. Verify the cabling between the PC and the EBC device network.
2. Verify the IP address given to the named device matches that of the actual device.
3. Verify that the device supports the requested address.

Unable to write to '<address>' on device '<device name>'

Error Type:

Serious

Possible Cause:

1. The Ethernet connection between the device and the host PC is broken.
2. The named device may have been assigned an incorrect IP address.
3. The requested address is not available in the device.

Solution:

1. Verify the cabling between the PC and the EBC device network.
2. Verify the IP address given to the named device matches that of the actual device.
3. Verify that the device supports the requested address.

Device address '<address>' contains a syntax error

Error Type:

Warning

Possible Cause:

1. A tag address contains one or more invalid characters.
2. Bit addressing notation conflicts with the assigned data type.

Solution:

Reenter the address in the client application.

Address '<address>' is out of range for the specified device or register

Error Type:

Warning

Possible Cause:

A tag address that has been specified dynamically references a location that is beyond the range of supported locations for this address type on the specified device. This range is generic and is designed to set a hard upper limit on locations for the specified address type. Each module will impose their own upper limit which is less than or equal to the generic upper limit.

Solution:

Verify that the address is correct; if it is not, re-enter it in the client application.

Device address '<address>' is not supported by model '<model name>'

Error Type:

Warning

Possible Cause:

A tag address that has been specified dynamically is not supported by the target model.

Solution:

Verify that the address is correct; if it is not, re-enter it in the client application. Verify the selected model name for the device is correct.

Device address '<address>' is Read Only

Error Type:

Warning

Possible Cause:

A tag address that has been specified dynamically has a requested access mode that is not compatible with what the device supports for that address.

Solution:

Change the access mode in the client application.

Data type '<type>' is not valid for device address '<address>'

Error Type:

Warning

Possible Cause:

A tag address that has been specified dynamically has been assigned an invalid data type.

Solution:

Modify the requested data type in the client application.

Device <device name> returned an error with value = <error value> disabling Link Watchdog. All tags will be invalidated.

Error Type:

Warning

Possible Cause:

The EBC device generated an error during a Link Watchdog disabling procedure. Due to this failure, it is uncertain whether the Link Watchdog is truly disabled on the device-side. All device tags will be invalidated for precautionary reasons.

Solution:

Such an error state cannot be cleared unless the problem is remedied. Check the error/warning/info list for a detailed description of this error value. If is not included in the list, contact the device manufacturer for further assistance.

See Also:

[EBC Error Codes](#)

Device <device name> returned an error with value = <error value> enabling Link Watchdog with timeout = <timeout>. All tags will be invalidated

Error Type:

Warning

Possible Cause:

The EBC device generated an error during a Link Watchdog enabling procedure. Due to this failure, it is uncertain whether the Link Watchdog is truly enabled on the device-side with the specified timeout. All tags will be invalidated for precautionary reasons.

Solution:

Such an error state cannot be cleared unless the problem is remedied. Check the error/warning/info list for a detailed description of this error value. If is not included in the list, contact the device manufacturer for further assistance.

See Also:

[EBC Error Codes](#)

Read Errors

The following error/warning messages may be generated. Click on the link for a description of the message.

[Address '<address>' is out of range for device '<device name>'. Tag Deactivated](#)

[Base referenced in address '<address>' on device '<device name>' does not exist. Tag Deactivated](#)
[Module referenced in address '<address>' on device '<device name>' does not exist. Tag Deactivated](#)
[Invalid address type for address '<address>' on device '<device name>'. Tag Deactivated](#)
[Device '<device name>' returned an error code with value = '<value>' reading address '<address>'](#)
[Device '<device name>' : Base '<base>' : Slot '<slot>' returned an error with value = '<error value>'](#)
[Device '<device name>' : Base '<base>' : Slot '<slot>' returned a warning with value = '<warning value>'](#)
[Device '<device name>' : Base '<base>' : Slot '<slot>' returned information with value = '<info value>'](#)
[Device '<device name>' : Base '<base>' : Slot '<slot>' returned an internal error with value = '<internal value>'](#)
[Device '<device name>' : Base '<base>' : Slot '<slot>' returned extended error: Type = '<ext err type>', Data = '<error bytes>'](#)
[Frame received from device '<device name>' contains errors](#)
[Frame received from device '<device name>' contains errors. Verify its IP address](#)
[Format version of frame received from device '<device name>' is not supported. Tag Deactivated](#)
[Model selected for device '<device name>' does not match the actual device model. Verify its IP address](#)
[Device '<device name>' block request \[<start> to <end>\] responded with exception <code>](#)
[Bad received length \[<start> to <end>\] on device '<device name>'](#)

Address '<address>' is out of range for device '<device name>'. Tag Deactivated

Error Type:

Warning

Possible Cause:

A tag address references a location that is considered out of range by the target module.

Solution:

Verify that the address is correct; if it is not, re-enter it in the client application.

Base referenced in address '<address>' on device '<device name>' does not exist. Tag Deactivated

Error Type:

Warning

Possible Cause:

A tag address references an extension base that does not exist. Applies to H4-EBC(-F).

Solution:

Verify that the address is correct; if it is not, re-enter it in the client application.

Module referenced in address '<address>' on device '<device name>' does not exist. Tag Deactivated

Error Type:

Warning

Possible Cause:

A tag address references a slot that is not occupied by an I/O module.

Solution:

Verify that the address is correct; if it is not, re-enter it in the client application.

Address type invalid for address '<address>' on device '<device name>'. Tag Deactivated

Error Type:

Warning

Possible Cause:

A tag references an address using an address type that is not valid for that module.

Solution:

Verify that the address type is correct; if it is not, re-enter it in the client application. Recall that 'X' (inputs) and 'Y' (outputs) are for discrete modules while 'K' (inputs) and 'V' (outputs) are for analog.

Device '<device name>' returned an error code with value = '<value>' reading address '<address>'

Error Type:

Warning

Possible Cause:

Device <device name> returned an error code during a read request to address <address>. Given the address is that of a non-blocking tag (ie. serial port tags), the error will be displayed for each non-block read and invalidated. If the address is that of a block read (ie. i/o tags), the address will be displayed as <I/O Block>. This means that the error and the invalidation of, applies to all tags for this device, excluding the serial port tags.

Solution:

Such an error state can not be cleared unless the problem is remedied. Check the error/warning/info list for a detailed description of this error value. If is not included in the list, contact the device manufacturer for further assistance.

See Also:

[EBC Error Codes](#)

Device '<device name>' : Base '<base>' : Slot '<slot>' returned an error with value = '<error value>'

Error Type:

Warning

Possible Cause:

The module in base <base>, slot <slot>, generated an error during its operation. All tags referencing this slot will be invalidated.

Solution:

Such an error state can not be cleared unless the problem is remedied. Check the error/warning/info list for a detailed description of this error value. If is not included in the list, contact the device manufacturer for further assistance.

See Also:

[EBC Error Codes](#)

Device '<device name>' : Base '<base>' : Slot '<slot>' returned a warning with value = '<warning value>'

Error Type:

Warning

Possible Cause:

The module in base <base>, slot <slot>, generated a warning during its operation.

Solution:

This warning state will be cleared if possible. Check the error/warning/info list for a detailed description of this warning

value. If is not included in the list, contact the device manufacturer for further assistance.

See Also:

[EBC Error Codes](#)

Device '<device name>' : Base '<base>' : Slot '<slot>' returned information with value = '<info value>'

Error Type:

Information

Possible Cause:

The module in base <base>, slot <slot>, generated information during its operation.

Solution:

This state will be cleared. Check the error/warning/info list for a detailed description of this information value. If is not included in the list, contact the device manufacturer for further assistance.

See Also:

[EBC Error Codes](#)

Device '<device name>' : Base '<base>' : Slot '<slot>' returned an internal error with value = '<internal value>'

Error Type:

Error

Possible Cause:

The module in base <base>, slot <slot>, generated an internal error during its operation. All tags referencing this slot will be invalidated.

Solution:

Such an error state is fatal and can not be cleared unless the problem is remedied. Check the error/warning/info list for a detailed description of this internal error value. If is not included in the list, contact the device manufacturer for further assistance.

See Also:

[EBC Error Codes](#)

Device '<device name>' : Base '<base>' : Slot '<slot>' returned extended error: Type = '<ext err type>', Data = '<error bytes>'

Error Type:

Warning

Possible Cause:

The module in base <base>, slot <slot>, generated an extended error of type <ext err type> during its operation. Extended error information is contained within the error bytes. All tags referencing this slot will be invalidated.

<error bytes> is a generic array of bytes. Its meaning depends on the module and the error. In most cases, each byte represents an analog channel.

00 = analog channel good

01 = analog channel in error. i.e. blown fuse, broken transmitter.

For example, given a thermocouple module:

"...Data = '00 00 01 00 00 00 00 00 00 00 01 00 00 00 00 00'" means Channel 3 and Channel 11 contain broken transmitters.

Solution:

Such an error state can not be cleared unless the problem is remedied. Check the error/warning/info list for a detailed description of this error value. If is not included in the list, contact the device manufacturer for further assistance.

See Also:

[EBC Error Codes](#)

Frame received from device '<device name>' contains errors

Error Type:

Warning

Possible Cause:

The EBC device <device name> responded with incorrect data possibly due to transmission errors or device malfunction.

Solution:

1. Place device on less noisy network if that is the case.
2. Increase the request timeout.

Frame received from device '<device name>' contains errors. Verify its IP address

Error Type:

Warning

Possible Cause:

The EBC device <device name> responded with incorrect data for the specified model but valid for another model.

Solution:

Verify the correct device model and IP address was selected.

Format version of frame received from device '<device name>' is not supported. Tag Deactivated

Error Type:

Warning

Possible Cause:

The EBC device <device name> responded with a frame whose protocol version is not supported. Currently, version 1 and below are supported.

Solution:

The AutomationDirect EBC driver will not be able to communicate with this device.

Model selected for device '<device name>' does not match the actual device model. Verify its IP address

Error Type:

Warning

Possible Cause:

Terminator I/O EBC's specify its model within the base definition. If a device specified in the server project is given a model that does not match the model specification in the base definition (ie. non-Terminator I/O), then this error will occur. All tags for this device will be invalidated.

Solution:

Verify the model and/or IP address selected for device <device name>.

Device '<device name>' block request [<start> to <end>] responded with exception <code>

Error Type:

Warning

Possible Cause:

Device <device name> returned an error code during a GS Drive block read request starting at address <start> and ending at address <end>. Addresses are listed in Modbus address form (4xxxx). Consult the drive's user manual for the corresponding parameters for the given start and end Modbus addresses. For example,

4-03 Analog Input Gain

Memory Address 0403H (41208)

In this example, the address 41208 is the Modbus address for parameter P4.3.

Solution:

Such an error state can not be cleared unless the problem is remedied. Contact technical support.

Bad received length [<start> to <end>] on device '<device name>'

Error Type:

Warning

Cause:

The driver attempted to read a block of memory in the GS Drive. The Drive responded with no error, but did not provide the driver with the requested block size of data.

Solution:

Ensure that the range of memory exists for the GS Drive.

Write Errors

The following error/warning messages may be generated. Click on the link for a description of the message.

[Write rejected. Address '<address>' is out of range for device '<device name>'](#)

[Write rejected. Base referenced in address '<address>' on device '<device name>' does not exist](#)

[Write rejected. Module referenced in address '<address>' on device '<device name>' does not exist](#)

[Write rejected. Invalid address type for address '<address>' on device '<device name>'](#)

[Write to address '<address>' failed. Device '<device name>' returned an undefined status code with value = '<value>'](#)

[Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned an error with value = '<error value>'](#)

[Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned a warning with value = '<warning value>'](#)

[Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned information with value = '<info value>'](#)

[Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned an internal error with value = '<internal value>'](#)

[Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned extended error: Type = '<ext err type>', Data = '<error bytes>'](#)

[Write to address '<address>' failed. Frame received from device '<device name>' contains errors](#)

[Write to address '<address>' failed. Frame received from device '<device name>' contains errors. Verify its IP address](#)

[Write to address '<address>' failed. Format version of frame received from device '<device name>' is not supported](#)

[Write to address '<address>' failed. Model selected for device '<device name>' does not match the actual device model. Verify its IP address](#)

[Unable to write to address <address>. Device '<device name>' responded with exception <code>](#)

Write rejected. Address '<address>' is out of range for device '<device name>'

Error Type:

Warning

Possible Cause:

A write operation was attempted on a location that is beyond the range of supported locations for the device.

Solution:

Verify that the address is correct; if it is not, re-enter it in the client application.

Write rejected. Base referenced in address '<address>' on device '<device name>' does not exist

Error Type:

Warning

Possible Cause:

A write operation was attempted on an extension base that does not exist. Applies to H4-EBC(-F).

Solution:

Verify that the address is correct; if it is not, re-enter it in the client application.

Write rejected. Module referenced in address '<address>' on device '<device name>' does not exist

Error Type:

Warning

Possible Cause:

A write operation was attempted on a slot that is not occupied by an I/O module.

Solution:

Verify that the address is correct; if it is not, re-enter it in the client application.

Write rejected. Invalid address type for address '<address>' on device '<device name>'

Error Type:

Warning

Possible Cause:

A write tag references an address using an address type that is not valid for that module.

Solution:

Verify that the address type is correct; if it is not, re-enter it in the client application. Recall that 'X' (inputs) and 'Y' (outputs) are for discrete modules while 'K' (inputs) and 'V' (outputs) are for analog.

Write to address '<address>' failed. Device '<device name>' returned an undefined status code with value = '<value>'

Error Type:

Warning

Possible Cause:

Device <device name> returned a status code that is not an error/warning/info/internal/extended error. Write failed.

Solution:

N/A.

Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned an error with value = '<error value>'**Error Type:**

Warning

Possible Cause:

The module in base <base>, slot <slot>, generated an error during a write operation to address <address>. Write failed.

Solution:

Such an error state can not be cleared unless the problem is remedied. Check the error/warning/info list for a detailed description of this error value. If is not included in the list, contact the device manufacturer for further assistance.

See Also:

[EBC Error Codes](#)

Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned a warning with value = '<warning value>'**Error Type:**

Warning

Possible Cause:

The module in base <base>, slot <slot>, generated a warning during a write operation to address <address>. Write may have succeeded.

Solution:

This warning state will be cleared if possible. Check the error/warning/info list for a detailed description of this warning value. If is not included in the list, contact the device manufacturer for further assistance.

See Also:

[EBC Error Codes](#)

Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned information with value = '<info value>'**Error Type:**

Information

Possible Cause:

The module in base <base>, slot <slot>, generated information during a write operation to address <address>. Write may have succeeded.

Solution:

This state will be cleared. Check the error/warning/info list for a detailed description of this information value. If is not included in the list, contact the device manufacturer for further assistance.

See Also:

[EBC Error Codes](#)

Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned an internal error with value = '<internal value>'

Error Type:

Error

Possible Cause:

The module in base <base>, slot <slot>, generated an internal error during a write operation to address <address>. Write failed.

Solution:

Such an error state is fatal and can not be cleared unless the problem is remedied. Check the error/warning/info list for a detailed description of this internal error value. If is not included in the list, contact the device manufacturer for further assistance.

See Also:

[EBC Error Codes](#)

Write to address '<address>' failed. Device '<device name>' : Base '<base>' : Slot '<slot>' returned extended error: Type = '<ext err type>', Data = '<error bytes>'

Error Type:

Warning

Possible Cause:

The module in base <base>, slot <slot>, generated an extended error of type <ext err type> during a write operation to address <address>. Extended error information is contained within the error bytes. Write failed.

Solution:

Such an error state can not be cleared unless the problem is remedied. Check the error/warning/info list for a detailed description of this error value. If is not included in the list, contact the device manufacturer for further assistance.

See Also:

[EBC Error Codes](#)

Write to address '<address>' failed. Frame received from device '<device name>' contains errors

Error Type:

Warning

Possible Cause:

A write operation was retried the preset number of times and failed each time. This is possibly due to transmission errors or device malfunction.

Solution:

1. Place device on less noisy network if that is the case.
2. Increase the request timeout.

Write to address '<address>' failed. Frame received from device '<device name>' contains errors. Verify its IP address

Error Type:

Warning

Possible Cause:

A write operation was retried the preset number of times and failed each time.

Solution:

Verify the correct device model and IP address was selected.

Write to address '<address>' failed. Format version of frame received from device '<device name>' is not supported

Error Type:

Warning

Possible Cause:

The EBC device <device name> responded with a frame whose protocol version is not supported. Currently, version 1 and below are supported. Write failed.

Solution:

The AutomationDirect EBC driver will not be able to communicate with this device.

Write to address '<address>' failed. Model selected for device '<device name>' does not match the actual device model. Verify its IP address

Error Type:

Warning

Possible Cause:

Terminator I/O EBC's specify its model within the base definition. If a device specified in the server project is given a model that does not match the model specification in the base definition (ie. non-Terminator I/O), then this error will occur. Write failed.

Solution:

Verify the model and/or IP address selected for device <device name>.

Unable to write to address <address>. Device '<device name>' responded with exception <code>

Error Type:

Warning

Possible Cause:

Device <device name> returned an error code during a GS Drive write request to address <address>. Addresses are listed in Modbus address form (4xxxx). Consult the drive's user manual for the corresponding parameters for the given start and end Modbus addresses. For example,

4-03 Analog Input Gain
Mem Addr 0403H (41208)

The address 41208 is the Modbus address for parameter P4.3.

Solution:

Such an error state can not be cleared unless the problem is remedied. Contact technical support.

Application Notes

Applications notes exist for the following modules. Select a link from the following list to obtain specific information for the module of interest.

[T1F-14THM](#)

[DF-HSC](#)

Application Notes: T1F-14THM

Enabling/Disabling of Thermocouple Channels

The 14THM Module provides 14 channels for thermocouple input. The module also provides the capability to enable/disable thermocouple channels based on the application's needs. It is preferred that users only enable the channels that will be used. An enabled channel without a thermocouple (or short to ground) will return a broken transmitter error to the AutomationDirect EBC I/O Device Driver.

Broken Transmitter = Enabled, open thermocouple channel.

All tags referencing channels with broken transmitters will have a bad quality. All tags referencing disabled channels will have a good quality since technically, disabled channels are not in error. The driver cannot distinguish good data originating from enabled channels from meaningless data originating from disabled channels. For this reason, it is important that users only reference enabled channels from the client.

There are two options for viewing the status of a Thermocouple Channel.

Event Log

The list of broken transmitters for a thermocouple module is given in the error "[Device '<device name>' : Base '<base>' : Slot '<slot>' returned extended error: Type = '<ext err type>', Data = '<error bytes>'](#)" where the <error bytes> lists the status of each channel in the form of an array. The meaning of each array byte is 00 = analog channel good, 01 = broken transmitter.

For example,

"...Data = '00 00 01 00 00 00 00 00 00 00 00 00 00 00 00'" means Channel 3 contains a broken transmitter.

Tag

Broken transmitter information can be referenced from a tag through [Terminator I/O Status Addressing](#). The syntax for referencing broken transmitters is S1:STS.EXT<nn> where nn is the 0-based Thermocouple Channel. The values of the tag are defined as

0 = analog channel good

1 = broken transmitter

Example

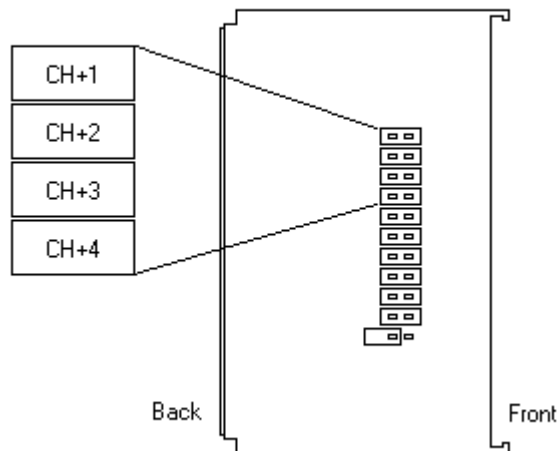
S1:STS.EXT0 = Extended Error for Thermocouple Channel 1 in Slot 1

S1:STS.EXT0 = 1 means a broken transmitter has been detected on Channel 1.

Note: If a channel is referenced that exceeds the number of channels for the module, STS.EXT<nn> will hold the value 0.

The diagram below is of the jumper layout for the 14THM module and a table for configuring the number of channels desired.

Top View of T1F-14THM Module Jumpers



Number of Channels Enabled	Jumper CH+1	Jumper CH+2	Jumper CH+3	Jumper CH+4
1				
2	X			
3		X		
4	X	X		
5			X	
6	X		X	
7		X	X	
8	X	X	X	
9				X
10	X			X
11		X		X
12	X	X		X
13			X	X
14	X	X	X	X

Power-Up Notes

Upon powering-up, it takes time for the 14THM module to detect broken transmitters. Between the time a broken transmitter is powered-up and detected, the 14THM module returns data based on radiant noise picked up by that open channel. For this reason, it is possible to receive valid data from a channel with a broken transmitter. Once a broken transmitter is detected, the EBC returns updated error information for that channel, which allows the driver to properly invalidate tags referencing that open channel.

Application Notes: D4-HSC

4 DWI (DWord In) locations share memory with the 4 DWO (DWord Out) locations as mapped below.

Mapping	Data
DWI1 <--> DWO0	Offset Value (4 bytes)
DWI2 <--> DWO1	Preset Value (4 bytes)
DWI3 <--> DWO2	Deceleration (4 bytes)
DWI5 <--> DWO3	Timebase (2 bytes but represented in 4 bytes)

These DWI and DWO locations have been designed to be Read Only and Write Only respectively. DWI is truly Read Only

from the OPC Server standpoint, but DWO has been enhanced to be read/write for maximum flexibility. In summary:

For	Read From	Write To
Offset Value	DWI1 or DWO0	DWO0
Preset Value	DWI2 or DWO1	DWO1
Deceleration	DWI3 or DWO2	DWO2
Timebase	DWI5 or DWO3	DWO3

Note: For further information on D4-HSC Data locations and usage, refer to the AutomationDirect D4-HSC Manual.

Index

- A -

Address '<address>' is out of range for the specified device or register. 27, 29
 Address Descriptions 9
 Address type invalid for address '<address>' on device '<device name>'. Tag Deactivated. 29
 Application Notes 37
 Application Notes D4-HSC 39
 Application Notes T1F-14THM 38

- B -

Base referenced in address '<address>' on device '<device name>' does not exist. Tag Deactivated. 29
 Boolean 8
 broken transmitter 38
 Byte 8

- C -

Cable Diagrams 6
 Communication Settings 6

- D -

D4-HSC 39
 Data Type 8
 Data type '<type>' is not valid for device address '<address>'. 28
 Data Types Description 8
 Device '<device name>'
 Base '<base>' : Slot '<slot>' returned a warning with value = '<warning value>'. 30
 Base '<base>' : Slot '<slot>' returned an error with value = '<error value>'. 30
 Base '<base>' : Slot '<slot>' returned an internal error with value = '<internal value>'. 31
 Base '<base>' : Slot '<slot>' returned extended error: Type = '<ext err type>' 31
 Base '<base>' : Slot '<slot>' returned extended error: Type = '<ext err type>':Data = '<error bytes>' 31

Base '<base>' : Slot '<slot>' returned information with value = '<info value>'. 31
 Device '<device name>' not responding. 26
 Device <device name> returned an error with value <error value> disabling Link Watchdog. All tags will be invalidated. 28
 Device <device name> returned an error with value <error value> enabling Link Watchdog with timeout <timeout>. All tags will b 28
 Device '<device name>' returned an undefined status code with value = '<value>'. 30
 Device address '<address>' contains a syntax error. 27
 Device address '<address>' is not supported by model '<model name>'. 27
 Device address '<address>' is read-only. 27
 Device ID 4
 Driver Error Messages 25
 Driver Warning Messages 26
 DWord 8

- E -

EBC 4
 EBC Error Codes 24
 EBC Error Descriptions 24
 EBC Errors 30, 31
 Error Descriptions 22
 Extended Error 9, 38

- F -

Float 8
 Flush Data Received 15
 Flush RX Queue 16
 Flush TX Queue 17
 Format version of frame received from device '<device name>' is not supported. 32
 Frame received from device '<device name>' contains errors. 32
 Frame received from device '<device name>' contains errors. Verify its IP address. 32

- G -

GS Drive 10, 20, 21
 GS Drive Parameter Addressing 20
 GS Drive Status Addressing 21
 GS1 20, 21

GS2 20, 21
GS3 20, 21

- H -

H2-EBC Addressing 10
H4-EBC Addressing 11

- I -

I/O Addressing 9
IP Address 4

- L -

Link Configuration 5
Long 8

- M -

Memory allocation error. 26
Model selected for device '<device name>' does not match the actual device model. Verify its IP address. 32
Module Hot Swapping 22
Module referenced in address '<address>' on device '<device name>' does not exist. 29

- N -

Network 4
Number of RX Queue Bytes 15
Number of TX Queue Bytes 17

- O -

Optimizing Your AutomationDirect EBC Communications 7
Overview 4

- P -

Parameter Addressing 10

- R -

Read Errors 28

- S -

Serial Data Received 15
Serial Data To Transmit 16
Serial Port Addresses 14
Serial Port Addressing 9, 14
Serial Port Configuration 14
 Baud Rate 17
 Data Bits 18
 Mode 19
 Parity 18
 Post-Transmit Delay 20
 Pre-Transmit Delay 19
 Stop Bits 18
 Use RTS Line 19
Short 8
Socket 25, 26
Status Addressing 9, 10

- T -

T1F-14THM 9, 38
Tag Deactivated 29
Terminator I/O 12
Thermocouple 9, 38

- U -

Unable to write tag '<address>' on device '<device name>.' 27

- W -

Winsock 25, 26
Winsock initialization failed (OS Error = n) 25
Winsock V1.1 or higher must be installed to use the AutomationDirect EBC device driver. 26
Word 8
Write 34, 36
Write Errors 33
Write failed. Frame received from device '<device name>' contains errors. 36

Write failed. Frame received from device '<device name>' contains errors. Verify its IP address. 36

Write rejected. Address '<address>' is out of range for device '<device name>'. 34

Write rejected. Base referenced in address '<address>' on device '<device name>' does not exist. 34

Write rejected. Invalid address type for address '<address>' on device '<device name>'. 34

Write rejected. Module referenced in address '<address>' on device '<device name>' does not exist. 34

Write to address '<address>' failed. Device '<device name>'

Base '<base>' : Slot '<slot>' returned a warning with value... 35

Base '<base>' : Slot '<slot>' returned an error with value = '<e 35

Base '<base>' : Slot '<slot>' returned an internal error with... 36

Base '<base>' : Slot '<slot>' returned extended error: Type... 36

Base '<base>' : Slot '<slot>' returned information with value... 35

Write to address '<address>' failed. Device '<device name>' returned an undefined status code with value = '<value>'. 34

Write to address '<address>' failed. Format version of frame received from device '<device name>' is not supported. 37

Write to address '<address>' failed. Model selected for device '<device name>' does not match the actual device model... 37