

# **Optimation OptiLogic Driver Help**

**© 2012 Kepware Technologies**

# Table of Contents

<b>Table of Contents</b> .....	<b>2</b>
<b>Optimization OptiLogic Driver Help</b> .....	<b>5</b>
<b>Overview</b> .....	<b>5</b>
<b>Device Setup</b> .....	<b>6</b>
<b>Optimizing Your Optimization OptiLogic Communications</b> .....	<b>8</b>
<b>Data Types Description</b> .....	<b>9</b>
<b>Address Descriptions</b> .....	<b>10</b>
<b>Digital I/O Bit Mapping</b> .....	<b>10</b>
<b>4 Digital Input Module</b> .....	<b>11</b>
<b>8 Digital Input Module</b> .....	<b>11</b>
<b>16 Digital Input Module</b> .....	<b>12</b>
<b>24 Digital Input Module</b> .....	<b>13</b>
<b>32 Digital Input Module</b> .....	<b>13</b>
<b>4 Digital Output Module</b> .....	<b>14</b>
<b>8 Digital Output Module</b> .....	<b>16</b>
<b>16 Digital Output Module</b> .....	<b>19</b>
<b>24 Digital Output Module</b> .....	<b>21</b>
<b>32 Digital Output Module</b> .....	<b>23</b>
<b>2 Channel Analog Input Module</b> .....	<b>25</b>
<b>4 Channel Analog Input Module</b> .....	<b>26</b>
<b>8 Channel Analog Input Module</b> .....	<b>26</b>
<b>16 Channel Analog Input Module</b> .....	<b>27</b>
<b>2 Channel Analog Output Module</b> .....	<b>27</b>
<b>4 Channel Analog Output Module</b> .....	<b>28</b>
<b>8 Channel Analog Output Module</b> .....	<b>29</b>
<b>16 Channel Analog Output Module</b> .....	<b>29</b>
<b>High Speed Counter Module</b> .....	<b>30</b>
<b>2 Channel High Speed Counter Module</b> .....	<b>35</b>
<b>Base RS232 Port</b> .....	<b>38</b>
<b>Dual RS232 Port Module</b> .....	<b>42</b>
<b>OL3406 Operator Panel</b> .....	<b>47</b>
<b>OL3420 Operator Panel</b> .....	<b>50</b>
<b>OL3440 Operator Panel</b> .....	<b>54</b>
<b>OL3850 Operator Panel</b> .....	<b>54</b>
<b>4 Digital Output Module</b> .....	<b>59</b>
<b>8 Digital Output Module</b> .....	<b>61</b>
<b>8 Digital Output Module</b> .....	<b>63</b>
<b>8 Digital Output Module</b> .....	<b>66</b>

8 Digital Input Module.....	68
4 Digital Input Module.....	68
8 Digital Input Module.....	69
8 Digital Input Module.....	70
2 Channel High Speed Counter Module.....	70
High Speed Counter Module.....	73
4 Channel Analog Output Module.....	79
8 Channel Analog Input Module.....	80
8 Channel Analog Input Module.....	80
Dual RS232 Port Module.....	81
OL3406 Operator Panel.....	85
OL3420 Operator Panel.....	89
OL3440 Operator Panel.....	92
OL3850 Operator Panel.....	93
Base RS232 Port.....	97
<b>Error Descriptions.....</b>	<b>103</b>
<b>OptiLogic Device Error Codes.....</b>	<b>104</b>
<b>Driver Error Messages.....</b>	<b>104</b>
Winsock initialization failed (OS Error = n).....	104
Winsock V1.1 or higher must be installed to use the OptiLogic device driver.....	104
Memory allocation error.....	104
<b>Driver Warning Messages.....</b>	<b>104</b>
Device '<device name>' is not responding.....	105
Device address <address> contains a syntax error.....	105
Address <address> is out of range for the specified device or register.....	105
Device address <address> is not supported by model <model name>.....	105
Device address <address> is Read Only.....	106
Array size is out of range for address <address>.....	106
Array support is not available for the specified address: <address>.....	106
Data type <type> is not valid for device address <address>.....	106
Base with type=<type> Major ver.=<major> Minor ver.=<minor> is currently not supported. Contact Technical Support.....	106
Module in slot <slot> w/ type=<type>, subtype=<subtype> is currently not supported. Contact Technical Support.....	107
Panel with type=<type>, subtype=<subtype> is currently not supported. Contact Technical Support.....	107
<b>Read Errors.....</b>	<b>107</b>
Frame received from device <device name> contains errors.....	107
Base module referenced in address <address> on device <device name> does not exist.....	108
Slot module referenced in address <address> on device <device name> does not exist.....	108
Panel module referenced in address <address> on device <device name> does not exist.....	108

Address <address> contains an invalid address type for RTU module, device <device name>.....	108
Address <address> contains an invalid address type for module <subtype>, slot <slot>, device <device name>.....	108
Address <address> contains an invalid address type for panel module <subtype>, device <device name>.....	109
Address <address> is out of range for module <subtype> in slot <slot>, device <device name>.....	109
Address <address> is out of range for panel module <subtype>, device <device name>.....	109
Port <port> of the base module returned an error with value = <error value>.....	109
Module <subtype> in slot <slot> returned an error with value = <error value>.....	109
Port <port> of module <subtype> in slot <slot> returned an error with value = <error value>.....	110
Panel module <subtype> returned an error with value = <error value>.....	110
<b>Write Errors</b> .....	110
Unable to write to <address> on device <device name>.....	111
Write failed. Frame received from device <device name> contains errors.....	111
Write rejected. Base module referenced in address <address> on device <device name> does not exist.....	111
Write rejected. Slot module referenced in address <address> on device <device name> does not exist.....	111
Write rejected. Panel module referenced in address <address> on device <device name> does not exist.....	112
Write rejected. Address <address> contains an invalid address type for RTU module, device <device name>.....	112
Write rejected. Address <address> contains an invalid address type for module <subtype>, slot <slot>, device <device name>.....	112
Write rejected. Address <address> contains an invalid address type for panel module <subtype>, device <device name>.....	112
Write rejected. Address <address> is out of range for module <subtype> in slot <slot>, device <device name>.....	112
Write rejected. Address <address> is out of range for panel module <subtype>, device <device name>.....	113
Write failed. Port <port> of the base module returned an error with value = <error value>.....	113
Write failed. Module <subtype> in slot <slot> returned an error with value = <error value>.....	113
Write failed. Port <port> of module <subtype> in slot <slot> returned an error with value = <error value>.....	113
Write failed. Panel module <subtype> returned an error with value = <error value>.....	114
<b>Index</b> .....	115

## Optimization OptiLogic Driver Help

---

Help version 1.012

### CONTENTS

#### [Overview](#)

What is the Optimization OptiLogic Driver?

#### [Device Setup](#)

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

#### [Optimizing Your Optimization OptiLogic Communications](#)

How do I get the best performance from the Optimization OptiLogic Driver?

#### [Data Types Description](#)

What data types does this driver support?

#### [Address Descriptions](#)

How do I address data locations?

#### [Error Descriptions](#)

What error messages does the Optimization OptiLogic Driver produce?

### Overview

---

The Optimization OptiLogic Driver provides an easy and reliable way to connect Optimization OptiLogic devices to OPC Client applications, including HMI, SCADA, Historian, MES, ERP and countless custom applications. It is intended for use with Optimization OptiLogic Series Remote Terminal Units, I/O Modules and Operator Panels.

## Device Setup

---

### Supported RTUs

OL4054  
OL4058

### Supported Modules

OL2208, OL2211, OL2201  
OL2108, OL2109, OL2111  
OL2408, OL2418  
OL2252  
OL3406, OL3420, OL3440, OL3850

### Communication Protocol

Ethernet using Winsock V1.1 or higher.

### Device IDs

The Device ID is used to specify the device IP address along with an OptiLogic Node Number on the Ethernet network. Device IDs are specified as either YYY.YYY.YYY.YYY:XXX. The YYY designates the device IP address: each YYY byte should be in the range of 0 to 255. The XXX designates the OptiLogic node number of the RTU and can be in the range of 0 to 99. This allows up to 99 devices to be defined on a given channel.

**Note:** Each device on the channel must be uniquely identified by its own IP address and Node ID. Users can use the OptiLogic Update Tool software supplied by Optimization to configure an RTU's IP address. Users can set the RTU Node ID using the address switches under the base unit cover.

### Precision

The default precision will be used when writing float data to the device.

For example, if 50.02 is written from the client, the float value received by the server could be 50.019999. The driver needs to know the precision that should be used for this value. Thus, a bit number (representing the precision) should be appended to the address. A bit number of three would send the previous example as 50.019 with a precision of 3.

### Configuration

The Optimization OptiLogic Driver automatically detects the location of each module in the RTU. While certain I/O modules are required to use Slot 0, placement of others is flexible. For more information, refer to the specific module's owner manual. The number of slots varies with RTU: 4 for OL4054 and 8 for OL4058. Only one operator panel can be connected per RTU.

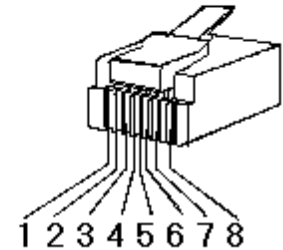
### Cable Diagrams

## 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	

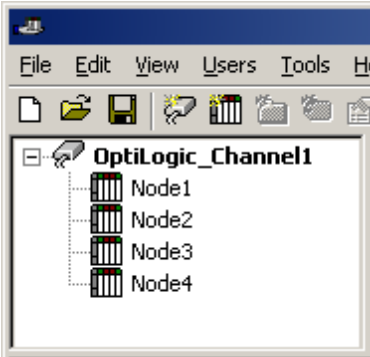
RJ45 RJ45

## 8-pin RJ45

## Optimizing Your Optimization OptiLogic Communications

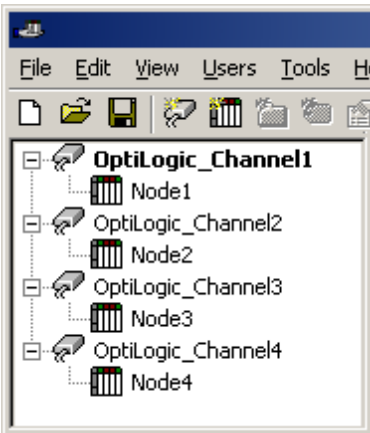
The Optimization OptiLogic driver has been designed to provide the best performance with the least amount of impact on the system's overall performance. While the Optimization OptiLogic driver is fast, there are a couple of guidelines that can be used in order to control and optimize the application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

Our server refers to communications protocols like Optimization OptiLogic 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 OptiLogic RTU 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 Optimization OptiLogic 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 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 Optimization OptiLogic driver could only define one single channel, then the example shown above would be the only option available; however, the Optimization OptiLogic 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 channels, it can be optimized exactly how it is shown here.

The performance will improve even if the application has more than 100 channels. While 100 or fewer channels 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
Double	64 bit floating point value
String	Null terminated character array

## Address Descriptions

---

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

### Select By SubType

[OL2104](#)  
[OL2205](#)  
[OL2304](#)  
[OL3420](#)  
[OL2108](#)  
[OL2208](#)  
[OL2408](#)  
[OL3440](#)  
[OL2109](#)  
[OL2211](#)  
[OL2418](#)  
[OL3850](#)  
[OL2111](#)  
[OL2252](#)  
[OL2602](#)  
[OL4054](#)  
[OL2201](#)  
[OL2258](#)  
[OL3406](#)

### Select By Classification

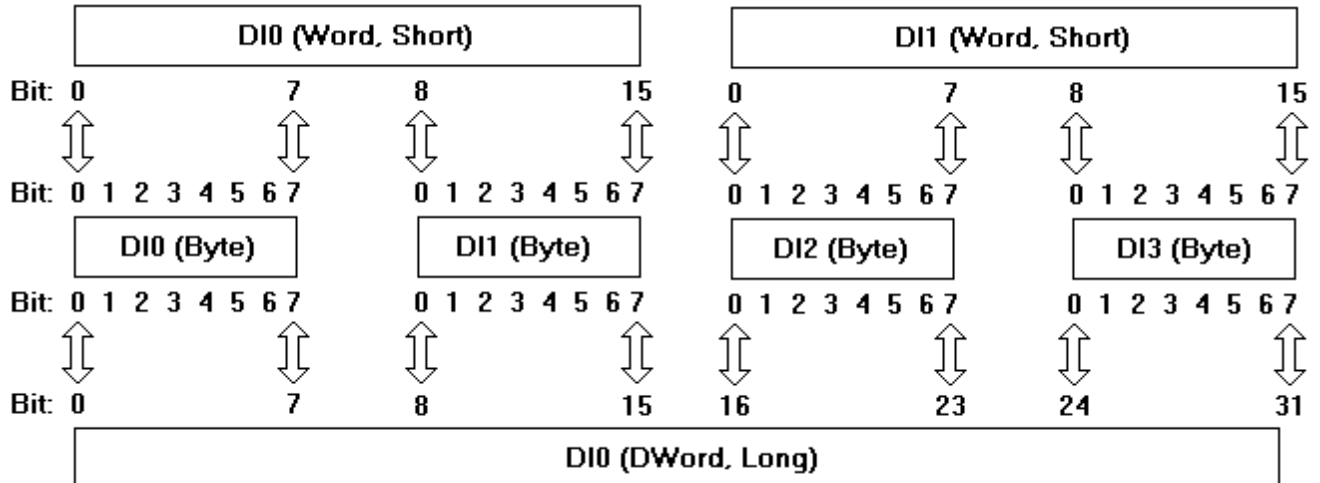
[4 Digital Input](#)  
[2 Channel Analog Input](#)  
[Base RS232 Port](#)  
[8 Digital Input](#)  
[4 Channel Analog Input](#)  
[Dual RS232 Port](#)  
[16 Digital Input](#)  
[8 Channel Analog Input](#)  
[OL3406 Operator Panel](#)  
[24 Digital Input](#)  
[16 Channel Analog Input](#)  
[OL3420 Operator Panel](#)  
[32 Digital Input](#)  
[2 Channel Analog Output](#)  
[OL3440 Operator Panel](#)  
[4 Digital Output](#)  
[4 Channel Analog Output](#)  
[OL3850 Operator Panel](#)  
[8 Digital Output](#)  
[8 Channel Analog Output](#)  
[16 Digital Output](#)  
[16 Channel Analog Output](#)  
[24 Digital Output](#)  
[High Speed Counter](#)  
[32 Digital Output](#)  
[2 Channel High Speed Counter](#)

This device requires a client connection: tag addresses cannot be validated completely until successful communication with the device is made. Since the driver is unaware of the device configuration until this communication is made, the tag addressing parser can only notify the user of improper addresses within the driver.

## Digital I/O Bit Mapping

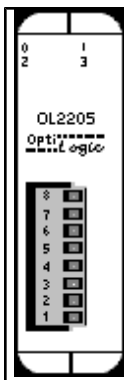
---

Points (bits) can be referenced using the optional Byte, Word, Short, DWord and Long data types. The following diagram illustrates how the driver maps bits within these data types.



All digital inputs and outputs can be addressed as Byte, Word, Short, DWord or Long data types. When addressing a module (such as an 8 bit input module) as a Word data type, the upper 8 bits will be fixed at zero while the lower bits will contain the status of the 8 inputs.

#### 4 Digital Input Module



Digital input modules are used to monitor the state of discrete field devices. Typical digital inputs are connected to switches, buttons, digital outputs from other equipment, discrete level sensors, thermostats and other On/Off sensing devices.

Digital status is sensed by a controller such as an OptiLogic system, by passing current through an input sensor. When the current is on, the input state is active. When there is no current, the state is inactive.

#### Subtypes

OL2205

#### 4 Digital Input Addressing Specifications

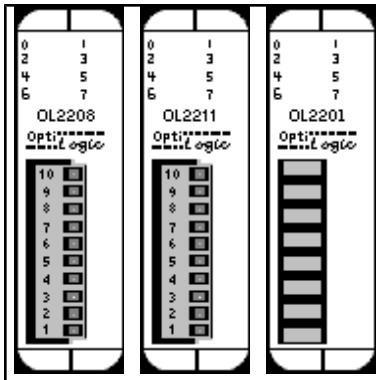
Syntax	Data Type	Range	Access
S<slot>:DI<point>	Boolean	0-3	Read Only
S<slot>:DI<offset>*	Byte	0	Read Only
S<slot>:DI<offset>*	Word, Short	0	Read Only
S<slot>:DI<offset>*	DWord, Long	0	Read Only

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

#### Examples

Address	Value	Description
S1:DI0	1	Slot 1, point 0 is on.
S1:DI0 (as Byte)	15	Slot 1, byte 0 (points 0-3 are on).
S1:DI0 (as Word)	0	Slot 1, word 0 (points 0-3 are off).
S1:DI0 (as DWORD)	15	Slot 1, DWord 0 (points 0-3 are on).

#### 8 Digital Input Module



Digital input modules are used to monitor the state of discrete field devices. Typical digital inputs are connected to switches, buttons, digital outputs from other equipment, discrete level sensors, thermostats and other On/Off sensing devices.

Digital status is sensed by a controller such as an OptiLogic system, by passing current through an input sensor. When the current is on, the input state is active. When there is no current, the state is inactive.

**Subtypes**

OL2208, OL2211, OL2201

**8 Digital Input Addressing Specifications**

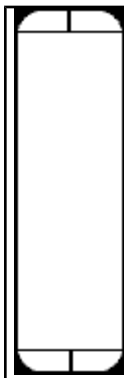
Syntax	Data Type	Range	Access
S<slot>:DI<point>	Boolean	0-7	Read Only
S<slot>:DI<offset>*	Byte	0	Read Only
S<slot>:DI<offset>*	Word, Short	0	Read Only
S<slot>:DI<offset>*	DWord, Long	0	Read Only

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

**Examples**

Address	Value	Description
S1:DI0	1	Slot 1, point 0 is on.
S1:DI0 (as Byte)	255	Slot 1, byte 0 (points 0-7 are on).
S1:DI0 (as Word)	0	Slot 1, word 0 (points 0-7 are off).
S1:DI0 (as DWORD)	255	Slot 1, DWord 0 (points 0-7 are on).

**16 Digital Input Module**



Digital input modules are used to monitor the state of discrete field devices. Typical digital inputs are connected to switches, buttons, digital outputs from other equipment, discrete level sensors, thermostats and other On/Off sensing devices.

Digital status is sensed by a controller such as an OptiLogic system, by passing current through an input sensor. When the current is on, the input state is active. When there is no current, the state is inactive.

**Subtypes**

N/A

**16 Digital Input Addressing Specifications**

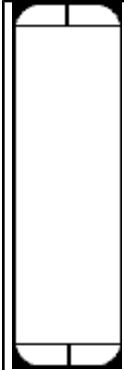
Syntax	Data Type	Range	Access
S<slot>:DI<point>	Boolean	0-7	Read Only
S<slot>:DI<offset>*	Byte	0-1	Read Only
S<slot>:DI<offset>*	Word, Short	0	Read Only
S<slot>:DI<offset>*	DWord, Long	0	Read Only

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

### Examples

Address	Value	Description
S1:DI0	1	Slot 1, point 0 is on.
S1:DI1 (as Byte)	255	Slot 1, byte 1 (points 8-15 are on).
S1:DI0 (as Word)	0	Slot 1, word 0 (points 0-15 are off).
S1:DI0 (as DWORD)	65535	Slot 1, DWord 0 (points 0-15 are on).

## 24 Digital Input Module



Digital input modules are used to monitor the state of discrete field devices. Typical digital inputs are connected to switches, buttons, digital outputs from other equipment, discrete level sensors, thermostats and other On/Off sensing devices.

Digital status is sensed by a controller such as an OptiLogic system, by passing current through an input sensor. When the current is on, the input state is active. When there is no current, the state is inactive.

### Subtypes

N/A

### 24 Digital Input Addressing Specifications

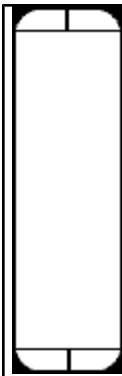
Syntax	Data Type	Range	Access
S<slot>:DI<point>	Boolean	0-7	Read Only
S<slot>:DI<offset>*	Byte	0-2	Read Only
S<slot>:DI<offset>*	Word, Short	0-1	Read Only
S<slot>:DI<offset>*	DWord, Long	0	Read Only

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

### Examples

Address	Value	Description
S1:DI0	1	Slot 1, point 0 is on.
S1:DI2 (as Byte)	255	Slot 1, byte 2 (points 16-23 are on).
S1:DI0 (as Word)	0	Slot 1, word 0 (points 0-15 are off).
S1:DI0 (as DWORD)	16777216	Slot 1, DWord 0 (points 0-23 are on).

## 32 Digital Input Module



Digital input modules are used to monitor the state of discrete field devices. Typical digital inputs are connected to switches, buttons, digital outputs from other equipment, discrete level sensors, thermostats and other On/Off sensing devices.

Digital status is sensed by a controller such as an OptiLogic system, by passing current through an input sensor. When the current is on, the input state is active. When there is no current, the state is inactive.

**Subtypes**

N/A

**32 Digital Input Addressing Specifications**

Syntax	Data Type	Range	Access
S<slot>:DI<point>	Boolean	0-7	Read Only
S<slot>:DI<offset>*	Byte	0-3	Read Only
S<slot>:DI<offset>*	Word, Short	0-1	Read Only
S<slot>:DI<offset>*	DWord, Long	0	Read Only

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

**Examples**

Syntax	Data Type	Range	Access
S<slot>:DI<point>	Boolean	0-7	Read Only
S<slot>:DI<offset>*	Byte	0-3	Read Only
S<slot>:DI<offset>*	Word, Short	0-1	Read Only
S<slot>:DI<offset>*	DWord, Long	0	Read Only

**4 Digital Output Module**



Digital outputs are used to turn loads on and off. Loads may be lights, motors, solenoids or any type of On/Off device.

Digital outputs in the OptiLogic series come in three types: relay, transistor and solid state relay. Each type has applications it is best suited for. For more information, refer to the module manual.

If there is a loss of communication with the host, the output module will enter a fail safe state. There are three types of states: fail safe to all outputs off, fail safe to a pattern and fail safe to last state. The time in which a fail safe state is entered after loss of communication can be delayed up to 25.5 seconds. If certain points need to be treated differently in fail safe mode, a pattern can be used to signify which points are turned on and which ones are turned off in fail safe mode.

**Subtypes**

OL2104

**Address Types**

[4 Digital Output](#)

[Fail Safe Type](#)

[Fail Safe Pattern](#)

[Fail Safe Time](#)

[Fail Safe Set](#)

#### 4 Digital Output Addressing Specifications

Syntax	Data Type	Range	Access
S<slot>:DO<point>	Boolean	0-3	Read/Write
S<slot>:DO<offset>*	Byte**	0	Read/Write
S<slot>:DO<offset>*	Word, Short**	0	Read/Write
S<slot>:DO<offset>*	DWord, Long**	0	Read/Write

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

\*\*Since only the lower nibble of the least significant byte is being used, any value entered above 15 will be cropped.

#### Examples

Address	Value	Description
S1:DO0	1	Slot 1, point 0 (turn point 0 on).
S1:DO0 (as Byte)	15	Slot 1, byte 0 (turn points 0-3 on).
S1:DO0 (as Word)	0	Slot 1, word 0 (turn all points off).
S1:DO0 (as DWord)	15	Slot 1, DWord 0 (turn points 0-3 on).

#### Fail Safe Type Addressing

There are three types of fail safe states: fail safe to all outputs off, fail safe to a pattern and fail safe to last state. These types may be specified by referencing FS.TYPE.

#### Values

- 1 = Fail safe to all outputs off
- 2 = Fail safe to the pattern contained in FS<point>.PATTERN
- 3 = Fail safe to last state

#### Requirements

None

**Important:** The value specified in FS.TYPE is not sent to the device until FS.SET is set.

#### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.TYPE	Byte, Word, Short, DWord, Long	N/A	Write Only

#### Fail Safe Pattern Addressing

Using a fail safe pattern, users can specify which points are set and which ones are cleared when in fail safe mode.

#### Values

- True = turn point on
- False = turn point off

#### Requirements

FS.TYPE must be set to 2 for device to fail safe to a pattern.

**Note:** If using Byte, Word, Short, DWord or Long data types, the bit pattern determines which points are turned on and which ones are turned off.

**Important:** The pattern specified in FS.PATTERN is not sent to the device until FS.SET is set.

#### Specifications

Syntax	Data Type	Range	Access
S<slot>FS<point>.PATTERN	Boolean	0-3	Write Only
S<slot>:FS<offset>.PATTERN*	Byte**	0	Write Only
S<slot>:FS<offset>.PATTERN*	Word, Short**	0	Write Only
S<slot>:FS<offset>.PATTERN*	DWord, Long**	0	Write Only

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

\*\*Since only the lower nibble of the least significant byte is being used, any value entered above 15 will be cropped.

### Examples

Address	Value	Description
S1:FS0.PATTERN	1	Slot 1, turn point 0 on in fail safe mode.*
S1:FS0.PATTERN (as Byte)	15	Slot 1, turn points 0-3 on in fail safe mode.*
S1:FS0.PATTERN (as Word)	0	Slot 1, turn points 0-3 off in fail safe mode.*
S1:FS0.PATTERN (as DWORD)	15	Slot 1, turn points 0-3 on in fail safe mode.*

\*See Notes and Requirements above.

### Fail Safe Time Addressing

After communication between the host and the RTU has been lost, a time delay may be introduced before fail safe mode is engaged for the output module. This time delay can be entered by referencing FS.TIME and is specified in tenths of seconds.

### Requirements

None

**Important:** The value specified in FS.TYPE is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.TIME	Byte, Word, Short, DWord, Long	N/A	Write Only

### Examples

Address	Value	Description
S1:FS.TIME	255	Slot 1, fail safe time delay = 25.5 seconds.
S1:FS.TIME	0	Slot 1, no delay.
S1:FS.TIME	1	Slot 1, fail safe time delay = .1 seconds.

### Fail Safe Set Addressing

For any of the fail safe configuration parameters (type, time and pattern) to be sent to the device, FS.SET must be set. Immediately after the parameters are sent, FS.SET will be reset.

### Values

True = Send fail safe configurations to device  
False = No action

### Requirements

None

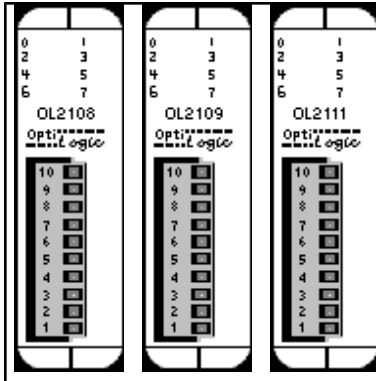
### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.SET	Boolean	N/A	Write Only

## 8 Digital Output Module

---





Digital outputs are used to turn loads on and off. Loads may be lights, motors, solenoids or any type of On/Off device.

Digital outputs in the OptiLogic series come in three types: relay, transistor and solid state relay. Each type has applications it is best suited for. For more information, refer to the module manual.

If there is a loss of communication with the host, the output module will enter a fail safe state. There are three types of states: fail safe to all outputs off, fail safe to a pattern and fail safe to last state. The time in which a fail safe state is entered after loss of communication can be delayed up to 25.5 seconds. If certain points need to be treated differently in fail safe mode, a pattern can be used to signify which points are turned on and which ones are turned off in fail safe mode.

### Subtypes

OL2108, OL2109, OL2111

### Address Types

#### [8 Digital Output](#)

#### [Fail Safe Type](#)

#### [Fail Safe Pattern](#)

#### [Fail Safe Time](#)

#### [Fail Safe Set](#)

### 8 Digital Output Addressing Specifications

Syntax	Data Type	Range	Access
S<slot>:DO<point>	Boolean	0-7	Read/Write
S<slot>:DO<offset>*	Byte	0	Read/Write
S<slot>:DO<offset>*	Word, Short	0	Read/Write
S<slot>:DO<offset>*	DWord, Long	0	Read/Write

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

### Examples

Address	Value	Description
S1:DO0	1	Slot 1, point 0 (turn point 0 on).
S1:DO0 (as Byte)	255	Slot 1, byte 0 (turn points 0-7 on).
S1:DO0 (as Word)	0	Slot 1, word 0 (turn points 0-7 off).
S1:DO0 (as DWord)	255	Slot 1, DWord 0 (turn points 0-7 on).

### Fail Safe Type Addressing

There are three types of fail safe states: fail safe to all outputs off, fail safe to a pattern and fail safe to last state. These types may be specified by referencing FS.TYPE.

### Values

- 1 = Fail safe to all outputs off
- 2 = Fail safe to the pattern contained in FS<point>.PATTERN
- 3 = Fail safe to last state

### Requirements

None

**Important:** The value specified in FS.TYPE is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.TYPE	Byte, Word, Short, DWord, Long	N/A	Write Only

### Fail Safe Pattern Addressing

Using a fail safe pattern, users can specify which points are set and which ones are cleared when in fail safe mode.

### Values

True = turn point on  
False = turn point off

### Requirements

FS.TYPE must be set to 2 for device to fail safe to a pattern.

**Note:** If using Byte, Word, Short, DWord or Long data types, the bit pattern determines which points are turned on and which ones are turned off.

**Important:** The pattern specified in FS.PATTERN is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>FS<point>.PATTERN	Boolean	0-7	Write Only
S<slot>:FS<offset>.PATTERN*	Byte	0	Write Only
S<slot>:FS<offset>.PATTERN*	Word, Short	0	Write Only
S<slot>:FS<offset>.PATTERN*	DWord, Long	0	Write Only

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

### Examples

Address	Value	Description
S1:FS0.PATTERN	1	Slot 1, turn point 0 on in fail safe mode.*
S1:FS0.PATTERN (as Byte)	255	Slot 1, turn points 0-7 on in fail safe mode.*
S1:FS0.PATTERN (as Word)	0	Slot 1, turn point 0-7 off in fail safe mode.*
S1:FS0.PATTERN (as DWORD)	255	Slot 1, turn points 0-7 on in fail safe mode.*

\*See Notes and Requirements above.

### Fail Safe Time Addressing

After communication between the host and the RTU has been lost, a time delay may be introduced before fail safe mode is engaged for the output module. This time delay can be entered by referencing FS.TIME and is specified in tenths of seconds.

### Requirements

None

**Important:** The value specified in FS.TYPE is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.TIME	Byte, Word, Short, DWord, Long	N/A	Write Only

### Examples

Address	Value	Description
S1:FS.TIME	255	Slot 1, fail safe time delay = 25.5 seconds.
S1:FS.TIME	0	Slot 1, no delay.
S1:FS.TIME	1	Slot 1, fail safe time delay = .1 seconds.

### Fail Safe Set Addressing

For any of the fail safe configuration parameters (type, time and pattern) to be sent to the device, FS.SET must be set. Immediately after the parameters are sent, FS.SET will be reset.

### Values

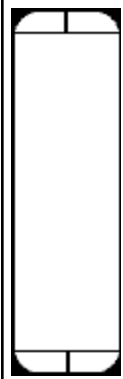
True = Send fail safe configurations to device  
False = No action

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:FS.SET	Boolean	N/A	Write Only

**16 Digital Output Module**

Digital outputs are used to turn loads on and off. Loads may be lights, motors, solenoids or any type of On/Off device.

Digital outputs in the OptiLogic series come in three types: relay, transistor and solid state relay. Each type has applications it is best suited for. For more information, refer to the module manual.

If there is a loss of communication with the host, the output module will enter a fail safe state. There are three types of states: fail safe to all outputs off, fail safe to a pattern and fail safe to last state. The time in which a fail safe state is entered after loss of communication can be delayed up to 25.5 seconds. If certain points need to be treated differently in fail safe mode, a pattern can be used to signify which points are turned on and which ones are turned off in fail safe mode.

**Subtypes**

N/A

**Address Types**[16 Digital Output](#)[Fail Safe Type](#)[Fail Safe Pattern](#)[Fail Safe Time](#)[Fail Safe Set](#)**16 Digital Output Addressing Specifications**

Syntax	Data Type	Range	Access
S<slot>:DO<point>	Boolean	0-7	Read/Write
S<slot>:DO<offset>*	Byte	0-1	Read/Write
S<slot>:DO<offset>*	Word, Short	0	Read/Write
S<slot>:DO<offset>*	DWord, Long	0	Read/Write

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

**Examples**

Address	Value	Description
S1:DO0	1	Slot 1, point 0 (turn point 0 on).
S1:DO1 (as Byte)	255	Slot 1, byte 1 (turn points 8-15 on).
S1:DO0 (as Word)	0	Slot 1, word 0 (turn points 0-15 off).
S1:DO0 (as DWord)	65535	Slot 1, DWord 0 (turn points 0-15 on).

**Fail Safe Type Addressing**

There are three types of fail safe states: fail safe to all outputs off, fail safe to a pattern and fail safe to last state. These types may be specified by referencing FS.TYPE.

**Values**

- 1 = Fail safe to all outputs off
- 2 = Fail safe to the pattern contained in FS<point>.PATTERN
- 3 = Fail safe to last state

**Requirements**

None

**Important:** The value specified in FS.TYPE is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.TYPE	<b>Byte</b> , Word, Short, DWord, Long*	N/A	Write Only

\*The default data type is shown in **bold**.

### Fail Safe Pattern Addressing

Using a fail safe pattern, users can specify which points are set and which ones are cleared when in fail safe mode.

### Values

True = turn point on  
False = turn point off

### Requirements

FS.TYPE must be set to 2 for device to fail safe to a pattern.

**Note:** If using Byte, Word, Short, DWord or Long data types, the bit pattern determines which points are turned on and which ones are turned off.

**Important:** The pattern specified in FS.PATTERN is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>FS<point>.PATTERN	<b>Boolean</b>	0-7	Write Only
S<slot>:FS<offset>.PATTERN*	Byte	0-1	Write Only
S<slot>:FS<offset>.PATTERN*	Word, Short	0	Write Only
S<slot>:FS<offset>.PATTERN*	DWord, Long	0	Write Only

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

### Examples

Address	Value	Description
S1:FS0.PATTERN	1	Slot 1, turn point 0 on in fail safe mode.*
S1:FS1.PATTERN (as Byte)	255	Slot 1, turn points 8-15 on in fail safe mode.*
S1:FS0.PATTERN (as Word)	0	Slot 1, turn points 0-15 on in fail safe mode.*
S1:FS0.PATTERN (as DWORD)	65535	Slot 1, turn points 0-15 on in fail safe mode.*

\*See Notes and Requirements above.

### Fail Safe Time Addressing

After communication between the host and the RTU has been lost, a time delay may be introduced before fail safe mode is engaged for the output module. This time delay can be entered by referencing FS.TIME and is specified in tenths of seconds.

### Requirements

None

**Important:** The value specified in FS.TYPE is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.TIME	<b>Byte</b> , Word, Short, DWord, Long	N/A	Write Only

### Examples

Address	Value	Description
---------	-------	-------------

S1:FS.TIME	255	Slot 1, fail safe time delay = 25.5 seconds.
S1:FS.TIME	0	Slot 1, no delay.
S1:FS.TIME	1	Slot 1, fail safe time delay = .1 seconds.

### Fail Safe Set Addressing

For any of the fail safe configuration parameters (type, time and pattern) to be sent to the device, FS.SET must be set. Immediately after the parameters are sent, FS.SET will be reset.

### Values

True = Send fail safe configurations to device  
False = No action

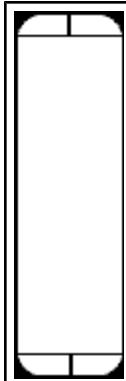
### Requirements

None

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.SET	Boolean	N/A	Write Only

## 24 Digital Output Module



Digital outputs are used to turn loads on and off. Loads may be lights, motors, solenoids or any type of On/Off device.

Digital outputs in the OptiLogic series come in three types: relay, transistor and solid state relay. Each type has applications it is best suited for. For more information, refer to the module manual.

If there is a loss of communication with the host, the output module will enter a fail safe state. There are three types of states: fail safe to all outputs off, fail safe to a pattern and fail safe to last state. The time in which a fail safe state is entered after loss of communication can be delayed up to 25.5 seconds. If certain points need to be treated differently in fail safe mode, a pattern can be used to signify which points are turned on and which ones are turned off in fail safe mode.

### Subtypes

N/A

### Address Types

[24 Digital Output](#)

[Fail Safe Type](#)

[Fail Safe Pattern](#)

[Fail Safe Time](#)

[Fail Safe Set](#)

### 24 Digital Output Addressing Specifications

Syntax	Data Type	Range	Access
S<slot>:DO<point>	Boolean	0-7	Read/Write
S<slot>:DO<offset>*	Byte	0-2	Read/Write
S<slot>:DO<offset>*	Word, Short	0-1	Read/Write
S<slot>:DO<offset>*	DWord, Long	0	Read/Write

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

### Examples

Address	Value	Description
S1:DO0	1	Slot 1, point 0 (turn point 0 on).
S1:DO2 (as Byte)	255	Slot 1, byte 2 (turn points 16-23 on).

S1:DO1 (as Word)	1	Slot 1, word 1 (turn point 16 on).
S1:DO0 (as DWORD)	16777216	Slot 1, DWord 0 (turn points 0-23 on).

### Fail Safe Type Addressing

There are three types of fail safe states: fail safe to all outputs off, fail safe to a pattern and fail safe to last state. These types may be specified by referencing FS.TYPE.

### Values

- 1 = Fail safe to all outputs off
- 2 = Fail safe to the pattern contained in FS<point>.PATTERN
- 3 = Fail safe to last state

### Requirements

None

**Important:** The value specified in FS.TYPE is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.TYPE	Byte, Word, Short, DWord, Long	N/A	Write Only

### Fail Safe Pattern Addressing

Using a fail safe pattern, users can specify which points are set and which ones are cleared when in fail safe mode.

### Values

- True = turn point on
- False = turn point off

### Requirements

FS.TYPE must be set to 2 for device to fail safe to a pattern.

**Note:** If using Byte, Word, Short, DWord or Long data types, the bit pattern determines which points are turned on and which ones are turned off.

**Important:** The pattern specified in FS.PATTERN is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>FS<point>.PATTERN	Boolean	0-7	Write Only
S<slot>:FS<offset>.PATTERN*	Byte	0-2	Write Only
S<slot>:FS<offset>.PATTERN*	Word, Short	0-1	Write Only
S<slot>:FS<offset>.PATTERN*	DWord, Long	0	Write Only

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#)

### Examples

Address	Value	Description
S1:FS0.PATTERN	1	Slot 1, turn point 0 on in fail safe mode.*
S1:FS2.PATTERN (as Byte)	255	Slot 1, turn points 16-23 on in fail safe mode.*
S1:FS1.PATTERN (as Word)	1	Slot 1, turn point 16 on in fail safe mode.*
S1:FS0.PATTERN (as DWORD)	16777216	Slot 1, turn points 0-23 on in fail safe mode.*

\*See Notes and Requirements above.

### Fail Safe Time Addressing

After communication between the host and the RTU has been lost, a time delay may be introduced before fail safe mode is engaged for the output module. This time delay can be entered by referencing FS.TIME and is specified in tenths of seconds.

### Requirements

None

**Important:** The value specified in FS.TYPE is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.TIME	Byte, Word, Short, DWord, Long	N/A	Write Only

### Examples

Address	Value	Description
S1:FS.TIME	255	Slot 1, fail safe time delay = 25.5 seconds.
S1:FS.TIME	0	Slot 1, no delay.
S1:FS.TIME	1	Slot 1, fail safe time delay = .1 seconds.

### Fail Safe Set Addressing

For any of the fail safe configuration parameters (type, time and pattern) to be sent to the device, FS.SET must be set. Immediately after the parameters are sent, FS.SET will be reset.

### Values

True = Send fail safe configurations to device  
False = No action

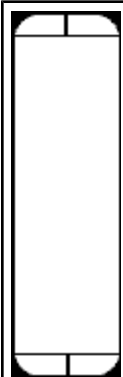
### Requirements

None

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.SET	Boolean	N/A	Write Only

## 32 Digital Output Module



Digital outputs are used to turn loads on and off. Loads may be lights, motors, solenoids or any type of On/Off device.

Digital outputs in the OptiLogic series come in three types: relay, transistor and solid state relay. Each type has applications it is best suited for. For more information, refer to the module manual.

If there is a loss of communication with the host, the output module will enter a fail safe state. There are three types of states: fail safe to all outputs off, fail safe to a pattern and fail safe to last state. The time in which a fail safe state is entered after loss of communication can be delayed up to 25.5 seconds. If certain points need to be treated differently in fail safe mode, a pattern can be used to signify which points are turned on and which ones are turned off in fail safe mode.

### Subtypes

N/A

### Address Types

[32 Digital Output](#)

[Fail Safe Type](#)

[Fail Safe Pattern](#)

[Fail Safe Time](#)

[Fail Safe Set](#)

### 32 Digital Output Addressing Specifications

Syntax	Data Type	Range	Access
S<slot>:DO<point>	Boolean	0-7	Read/Write
S<slot>:DO<offset>*	Byte	0-3	Read/Write

S<slot>:DO<offset>*	Word, Short	0-1	Read/Write
S<slot>:DO<offset>*	DWord, Long	0	Read/Write

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

### Examples

Address	Value	Description
S1:DO0	1	Slot 1, point 0 (turn point 0 on).
S1:DO0 (as Byte)	255	Slot 1, byte 0 (turn points 0-7 on).
S1:DO1 (as Word)	1	Slot 1, word 1 (turn point 16 on).
S1:DO0 (as DWORD)	4294967295	Slot 1, DWord 0 (turn points 0-31 on).

### Fail Safe Type Addressing

There are three types of fail safe states: fail safe to all outputs off, fail safe to a pattern and fail safe to last state. These types may be specified by referencing FS.TYPE.

### Values

- 1 = Fail safe to all outputs off
- 2 = Fail safe to the pattern contained in FS<point>.PATTERN
- 3 = Fail safe to last state

### Requirements

None

**Important:** The value specified in FS.TYPE is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.TYPE	Byte, Word, Short, DWord, Long	N/A	Write Only

### Fail Safe Pattern Addressing

Using a fail safe pattern, users can specify which points are set and which ones are cleared when in fail safe mode.

### Values

- True = turn point on
- False = turn point off

### Requirements

FS.TYPE must be set to 2 for device to fail safe to a pattern.

**Note:** If using Byte, Word, Short, DWord or Long data types, the bit pattern determines which points are turned on and which ones are turned off.

**Important:** The pattern specified in FS.PATTERN is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>FS<point>.PATTERN	Boolean	0-7	Write Only
S<slot>:FS<offset>.PATTERN*	Byte	0-3	Write Only
S<slot>:FS<offset>.PATTERN*	Word, Short	0-1	Write Only
S<slot>:FS<offset>.PATTERN*	DWord, Long	0	Write Only

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

### Examples

Address	Value	Description
S1:FS0.PATTERN	1	Slot 1, turn point 0 on in fail safe mode.*
S1:FS0.PATTERN (as Byte)	255	Slot 1, turn points 0-7 on in fail safe mode.*



S1:FS1.PATTERN (as Word)	1	Slot 1, turn point 16 on in fail safe mode.*
S1:FS0.PATTERN (as DWORD)	4294967295	Slot 1, turn points 0-31 on in fail safe mode.*

\*See Notes and Requirements above

### Fail Safe Time Addressing

After communication between the host and the RTU has been lost, a time delay may be introduced before fail safe mode is engaged for the output module. This time delay can be entered by referencing FS.TIME and is specified in tenths of seconds.

### Requirements

None

**Important:** The value specified in FS.TYPE is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.TIME	Byte, Word, Short, DWord, Long	N/A	Write Only

### Examples

Address	Value	Description
S1:FS.TIME	255	Slot 1, fail safe time delay = 25.5 seconds.
S1:FS.TIME	0	Slot 1, no delay.
S1:FS.TIME	1	Slot 1, fail safe time delay = .1 seconds.

### Fail Safe Set Addressing

For any of the fail safe configuration parameters (type, time and pattern) to be sent to the device, FS.SET must be set. Immediately after the parameters are sent, FS.SET will be reset.

### Values

True = Send fail safe configurations to device  
False = No action

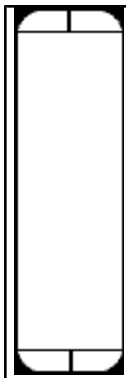
### Requirements

None

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.SET	Boolean	N/A	Write Only

## 2 Channel Analog Input Module



Analog inputs are used to monitor the value of continuously variable field measurements. Typical analog inputs are measurements of temperature, pressure, weight, liquid level, pH, flow rate and many other "real world" parameters.

Each channel of an analog input module takes an analog signal as input and uses a 12-bit A/D converter to put that analog value in digital form.

### Subtypes

N/A

### 2 Channel Analog Input Addressing Specifications

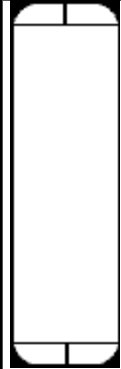
Syntax	Data Type	Range	Access
S<slot>:AI<channel>	Word, Short, DWord, Long	1-2	Read Only

**Examples**

Address	Value	Description
S1:AI1	*	Slot 1, channel 1.
S1:AI2	*	Slot 1, channel 2.

\*Value depends on input level, voltage/current ranges, accuracy and so forth.

**4 Channel Analog Input Module**



Analog inputs are used to monitor the value of continuously variable field measurements. Typical analog inputs are measurements of temperature, pressure, weight, liquid level, pH, flow rate and many other "real world" parameters.

Each channel of an analog input module takes an analog signal as input and uses a 12 bit A/D converter to put that analog value in digital form.

**Subtypes**

N/A

**4 Channel Analog Input Addressing Specifications**

Syntax	Data Type	Range	Access
S<slot>:AI<channel>	Word, Short, DWord, Long	1-4	Read Only

**Examples**


Address	Value	Description
S1:AI2	*	Slot 1, channel 2.
S1:AI4	*	Slot 1, channel 4.

\*Value depends on input level, voltage/current ranges, accuracy and so forth.

**8 Channel Analog Input Module**

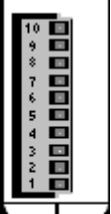
8 Channel  
Voltage In

OL2408



8 Channel  
Current In

OL2418



Analog inputs are used to monitor the value of continuously variable field measurements. Typical analog inputs are measurements of temperature, pressure, weight, liquid level, pH, flow rate and many other "real world" parameters.

Each channel of an analog input module takes an analog signal as input and uses a 12-bit A/D converter to put that analog value in digital form.

**Subtypes**

OL2408, OL2418

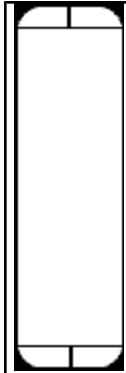
**8 Channel Analog Input Addressing Specifications**

Syntax	Data Type	Range	Access
S<slot>:AI<channel>	Word, Short, DWord, Long	1-8	Read Only

**Examples**

Address	Value	Description
S1:AI2	*	Slot 1, channel 2.
S1:AI8	*	Slot 1, channel 8.

\*Values depend on input level, voltage/current ranges, accuracy and so forth.

**16 Channel Analog Input Module**

Analog inputs are used to monitor the value of continuously variable field measurements. Typical analog inputs are measurements of temperature, pressure, weight, liquid level, pH, flow rate and many other "real world" parameters.

Each channel of an analog input module takes an analog signal as input and uses a 12-bit A/D converter to put that analog value in digital form.

**Subtypes**

N/A

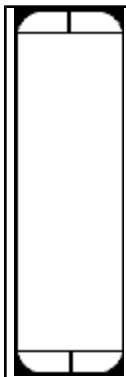
**16 Channel Analog Input Addressing Specifications**

Syntax	Data Type	Range	Access
S<slot>:AI<channel>	Word, Short, DWord, Long	1-16	Read Only

**Examples**

Address	Value	Description
S1:AI2	*	Slot 1, channel 2.
S1:AI16	*	Slot 1, channel 16.

\*Values depend on input level, voltage/current ranges, accuracy and so forth.

**2 Channel Analog Output Module**

Each channel of an analog output module uses a 12 bit D/A converter to produce an analog output signal.

**Subtypes**

N/A

**2 Channel Analog Output Addressing Specifications**

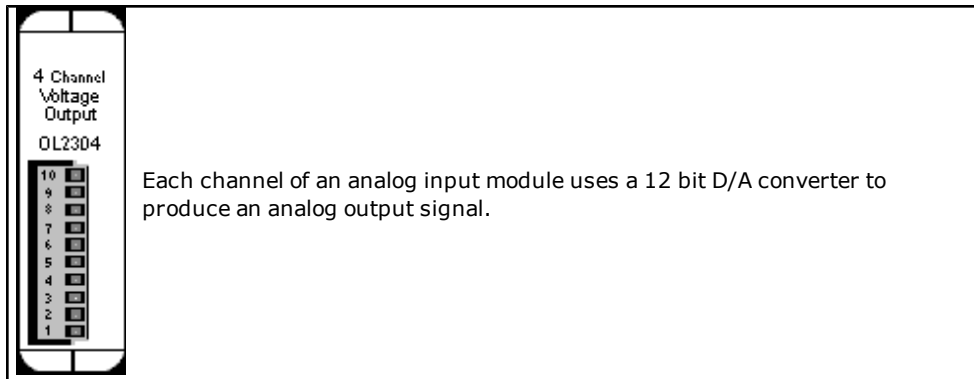
Syntax	Data Type	Range	Access
S<slot>:AO<channel>	Word, Short, DWord, Long	1-2	Write Only

### Examples

Address	Value	Description
S1:A01	1024 *	Slot 1, channel 1.
S1:A02	0 *	Slot 1, channel 2.

\*The exact analog output value depends on voltage range, accuracy and so forth.

## 4 Channel Analog Output Module



### Subtypes

OL2304

### Address Types

[4 Channel Analog Output](#)

[4 Channel Analog Output Range](#)

### 4 Channel Analog Output Addressing Requirements

As a precaution, the analog output range should be set prior to writing to an analog output channel.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:AO<channel>	Word, Short, DWord, Long	1-4	Write Only

### Examples

Address	Value	Description
S1:A02	1024 *	Slot 1, channel 2.
S1:A04	0 *	Slot 1, channel 4.

\*The exact analog output value depends on voltage range, accuracy and so forth.

### 4 Channel Analog Output Range Addressing

Each channel must be configured for a specific analog output range. This is done by setting AO<channel>.RANGE.

### Values

- 0 = 0-5 V
- 1 = 0-10 V
- 2 = +/- 5 V
- 3 = +/- 10 V

### Requirements

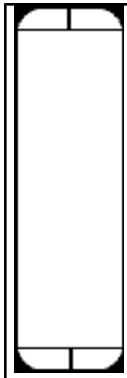
None.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:AO<channel>.RANGE	Word, Short, DWord, Long	1-4	Write Only

**Examples**

Address	Value	Description
S1:A02.RANGE	2	Slot 1, channel 2, set voltage range to +/- 5V.
S1:A04.RANGE	0	Slot 1, channel 4, set voltage range to 0-5V.

**8 Channel Analog Output Module**

Each channel of an analog input module uses a 12 bit D/A converter to produce an analog output signal.

**Subtypes**

N/A

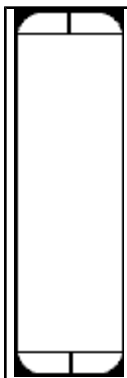
**8 Channel Analog Output Addressing Specifications**

Syntax	Data Type	Range	Access
S<slot>:AO<channel>	Word, Short, DWord, Long	1-8	Write Only

**Examples**

Address	Value	Description
S1:A02	1024 *	Slot 1, channel 2.
S1:A08	0 *	Slot 1, channel 8.

\*The exact analog output value depends on voltage range, accuracy and so forth.

**16 Channel Analog Output Module**

Each channel of an analog input module uses a 12-bit D/A converter to produce an analog output signal.

**Subtypes**

N/A

**16 Channel Analog Output Addressing Specifications**

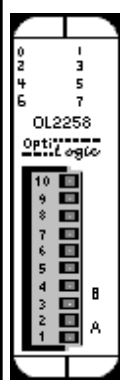
Syntax	Data Type	Range	Access
S<slot>:AO<channel>	Word, Short, DWord, Long	1-16	Write Only

### Examples

Address	Value	Description
S1:A02	1024 *	Slot 1, channel 2.
S1:A016	0 *	Slot 1, channel 16.

\*The exact analog output value depends on voltage range, accuracy and so forth.

## High Speed Counter Module



The OL2258 High Speed Pulse Counter module provides for direct pulse counting for a variety of high-speed pulse interface applications. Typical applications include motion control, metering and velocity measurement.

The OL2258 can be configured to operate in one of three modes:

1. Pulse & Direction (up to 80 kHz input).
2. Up/Down Count (up to 80 kHz input).
3. Quadrature (up to 160 kHz input).

Besides the counter's current count value, the pulse count in the most recent frequency period (user configurable) is also accessible. The OL2258 also contains 2 digital outputs, triggered on when the pulse count reaches the output's minimum range and triggered off when it reaches the output's maximum range.

### Subtypes

OL2258

### Address Types

[Channel Pulse Count](#)

[Channel Frequency Data](#)

[Channel Status: A](#)

[Channel Status: B](#)

[Channel Status: Z](#)

[Channel Status: LS](#)

[Channel Configuration: Count Type](#)

[Channel Configuration: Frequency Period](#)

[Channel Configuration: Preset](#)

[Channel Configuration: Force Preset](#)

[Channel Configuration: Hold Count](#)

[Channel Configuration: Z Preset Enable](#)

[Channel Configuration: LS Preset Enable](#)

[Triggered Digital Outputs](#)

[Triggered Digital Outputs: Minimum Range](#)

[Triggered Digital Outputs: Maximum Range](#)

[Triggered Digital Outputs: Range Enable](#)

### Channel Pulse Count

The pulse count for each channel can be referenced using address type C<channel>.COUNT.

### Requirements

None

### Specifications

Syntax	Data Type	Range	Access
S<slot>:C<channel>.COUNT	DWord, Long	1	Read Only

### Examples

Address	Value	Description
S1:C1.COUNT	0	Slot 1, channel 1 pulse count is 0.
S1:C1.COUNT	1000	Slot 1, channel 1 pulse count is 1000.

### Channel Frequency Data

A pulse count over a preset period of time (C<channel>.COUNT @ (start + period)-C<channel>.COUNT @ start) can be accessed through C<channel>.FREQDATA.

### Requirements

C<channel>.FREQPER determines the period and should be set before accessing C<channel>.FREQDATA.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:C<channel>.FREQDATA	Word, Short, DWord, Long	1	Read Only

### Examples

Address	Value	Description
S1:C1.FREQDATA	60000	Slot 1, ch1, if FREQPER = 1 sec, pulse is 60kHz.
S1:C1.FREQDATA	16000	Slot 1, ch1, if FREQPER = 200ms, pulse is 80kHz.

### Channel Status: A

Pulse input A can be referenced through CSTS<channel>.A. Count type defines the meaning of this input as follows:

Count Type	Input Definition
Pulse & Direction	Pulse Input
Up/Down Count	Up Pulse Input
Quadrature Encoder Input	Pulse Input

**Note:** For more information, refer to OptiLogic I/O Modules Manual.

### Values

True = Pulse level high  
False = Pulse level low

### Requirements

CCFG<channel>.COUNTTYPE determines the meaning of this input and should be set before accessing CSTS<channel>.A

### Specifications

Syntax	Data Type	Range	Access
S<slot>:CSTS<channel>.A	Boolean	1	Read Only

### Channel Status: B

Pulse input B can be referenced through CSTS<channel>.B. Count type defines the meaning of this input as follows:

Count Type	Input Definition
Pulse & Direction	Direction Input
Up/Down Count	Down Pulse Input
Quadrature Encoder Input	Pulse Input

**Note:** For more information, refer to OptiLogic I/O Modules Manual.

### Values

True = Pulse level high  
False = Pulse level low

### Requirements

CCFG<channel>.COUNTTYPE determines the meaning of this input and should be set before accessing CSTS<channel>.B

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CSTS<channel>.B	Boolean	1	Read Only

**Channel Status: Z**

Optional input Z provides a means of automatically resetting the count value to a user-defined preset value. Reference Z through CSTS<channel>.Z.

**Values**

True = If ZPRESETEN set, send PRESET to device. Otherwise, take no action.  
False = No action

**Requirements**

CCFG<channel>.ZPRESETEN must be set in order to enable use of Z in presetting counter

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CSTS<channel>.Z	Boolean	1	Read Only

**Channel Status: LS**

Optional limit switch input LS provides a means of automatically resetting the count value to a user-defined preset value. Reference LS through CSTS<channel>.LS.

**Values**

True = If LSPRESETEN set, send PRESET to device. Otherwise, take no action.  
False = No action

**Requirements**

CCFG<channel>.LSPRESETEN must be set in order to enable use of LS in presetting counter

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CSTS<channel>.LS	Boolean	1	Read Only

**Channel Configuration: Count Type (Mode)**

The mode of counter operation is configured via CCFG<channel>.COUNTTYPE.

**Values**

0 = Pulse & Direction  
1 = Up/Down Count  
2 = Quadrature

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.COUNTTYPE	Byte, Word, Short, DWord, Long	1	Read/Write

**Note:** For more information, refer to OptiLogic I/O Modules Manual.

**Examples**

Address	Value	Description
S1:CCFG1.COUNTTYPE	1	Slot 1, ch1 count mode set for pulse & dir.
S1:CCFG1.COUNTTYPE	3	Slot 1, ch1 count mode set for quadrature.

**Channel Configuration: Frequency Period**

Recall that a pulse count over a preset period of time can be accessed through C<channel>.FREQDATA. The period in the FREQDATA formula  $C<channel>.COUNT @ (start + period) - C<channel>.COUNT @ start$  is configured through CCFG<channel>.FREQPER.



**Values**

0 = 1 second count  
1 = 200 msec count

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.FREQPER	Byte, Word, Short, DWord, Long	1	Read/Write

**Examples**

Address	Value	Description
S1:CCFG1.FREQPER	0	Slot 1, ch1, FREQDATA reflects count in 1 second time window.
S1:CCFG1.FREQPER	1	Slot 1, ch1, FREQDATA reflects count in 200 ms time window.

**Channel Configuration: Preset**

Establish a preset count value by setting CCFG<channel>.PRESET.

**Requirements**

For preset to take effect, at least one of the following must be true:  
CCFG<channel>.FORCEPRESET be set to TRUE  
CCFG<channel>.ZPRESETEN be set to TRUE and Z input be TRUE  
CCFG<channel>.LSPRESETEN be set to TRUE and LS input be TRUE

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.PRESET	DWord, Long	1	Write Only

**Examples**

Address	Value	Description
S1:CCFG1.PRESET	1000	Slot 1, ch1, set preset to 1000.
S1:CCFG1.PRESET	0	Slot 1, ch1, set preset to 0 (default).

**Channel Configuration: Force Preset**

Set the counter's count value to the preset given in CCFG<channel>.PRESET.

**Values**

True = Send PRESET to device  
False = No action

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.FORCEPRESET	Boolean	1	Write Only

**Channel Configuration: Hold Count**

Hold the current count value by referencing CCFG<channel>.HOLDCOUNT.

**Values**

True = Hold count value  
False = No action

**Requirements**

None.

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.HOLDCOUNT	Boolean	1	Write Only

**Channel Configuration: Z Preset Enable**

Enables Z input to control the presetting of the counter.

**Values**

True = Z input state configured to preset counter when applicable  
False = Z input state has no control over presetting counter

**Requirements**

None.

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.ZPRESETEN	Boolean	1	Write Only

**Channel Configuration: LS Preset Enable**

Enables LS input to control the presetting of the counter.

**Values**

True = LS input state configured to preset counter when applicable  
False = LS input state has no control over presetting counter

**Requirements**

None.

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.LSPRESETEN	Boolean	1	Write Only

**Triggered Digital Outputs**

There are 2 digital outputs that can be configured to turn on when the counter count is within a specific range. The status of whether the digital output is on or off can be accessed through CTRIGDO<output>.

**Requirements**

CTRIGDO outputs must have a range specified and enabled, otherwise outputs will remain off.

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CTRIGDO<output>	Boolean	1-2	Read Only

**Examples**

Address	Value	Description
S1:CTRIGDO1	1	Slot 1, output 1 is on.
S1:CTRIGDO2	0	Slot 1, output 2 is off.

**Triggered Digital Outputs: Minimum Range**

Set the range's minimum value to be the counter's count value at which users desire the CTRIGDO<output> to turn from off to on. This minimum range value is referenced as CTRIGDO<output>.MINRANGE.

**Requirements**

CTRIGDO range usage must be enabled.

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CTRIGDO<output>.MINRANGE	DWord, Long	1-2	Write Only

**Examples**

Address	Value	Description
---------	-------	-------------

S1:CTRIGDO1.MINRANGE	100	Slot 1, output 1, turn on when count = 100.
S1:CTRIGDO2.MINRANGE	65536	Slot 1, output 2, turn on when count = 65536.

### Triggered Digital Outputs: Maximum Range

Set the range's maximum value to be the counter's count value at which users desire the CTRIGDO<output> to turn from on to off. This maximum range value is referenced as CTRIGDO<output>.MAXRANGE.

### Requirements

CTRIGDO usage range must be enabled.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:CTRIGDO<output>.MAXRANGE	DWord, Long	1-2	Write Only

### Examples

Address	Value	Description
S1:CTRIGDO1.MAXRANGE	1000	Slot 1, output 1, turn off when count = 1000.
S1:CTRIGDO2.MAXRANGE	0	Slot 1, output 2, turn off when count = 0 (from rollover).

### Triggered Digital Outputs: Range Enable

Enable the minimum and maximum range values for CTRIGDO (enable usage of a CTRIGDO output) by referencing CTRIGDO<output>.RANGEEN.

### Requirements

None.

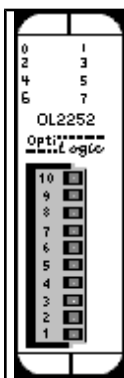
### Specifications

Syntax	Data Type	Range	Access
S<slot>:CTRIGDO<output>.RANGEEN	Boolean	1-2	Write Only

### Examples

Address	Value	Description
S1:CTRIGDO1.RANGEEN	1	Slot 1, output 1 enabled.
S1:CTRIGDO2.RANGEEN	0	Slot 1, output 2 disabled.

## 2 Channel High Speed Counter Module



The OL2252 Dual High Speed Pulse Counter module has two 0-15 kHz pulse counter inputs. Each input is independent of the other. There are a number of configuration options available. See OL2252 manual for details. Of the 10 inputs, 6 can be configured for general-purpose input and is referred to in this driver as CDI (counter digital input). Both channels have a software reset and an optional hardware reset as well as a software enable and optional hardware enable.

### Subtypes

OL2252

### Address Types

[Channel Pulse Count](#)

[Channel Enable \(Software\)](#)

[Channel Reset \(Software\)](#)

[Channel Enable \(Hardware\)](#)

[Channel Reset \(Hardware\)](#)

## [Channel Debounce Count](#) [8 Digital Input](#)

### Channel Pulse Count Addressing

The pulse count for each channel can be referenced using address type C<channel>.COUNT.

### Requirements

Channel counter must be enabled either by software or hardware for counter to pulse count.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:C<channel>.COUNT	DWord, Long	1-2	Read Only

### Examples

Address	Value	Description
S1:C1.COUNT	0	Slot 1, channel 1 pulse count is 0.*
S1:C2.COUNT	1000	Slot 1, channel 2 pulse count is 1000.*

\*See Requirements above.

### Channel Enable (Software) Addressing

Channels can be enabled/disabled through software by referencing address type C<channel>.EN.

### Values

True = channel enabled  
False = channel disabled\*

\*Provided channel is not currently hardware enabled.

### Requirements

None

### Specifications

Address	Value	Description
S1:C1.COUNT	0	Slot 1, channel 1 pulse count is 0.*
S1:C2.COUNT	1000	Slot 1, channel 2 pulse count is 1000.*

### Examples

Address	Value	Description
S1:C1.EN	0	Slot 1, ch1 not software enabled.
S1:C2.EN	1	Slot 1, ch2 software enabled.

### Channel Reset (Software) Addressing

A channel's pulse count can be reset through software by referencing address type C<channel>.RES.

### Values

True = channel pulse count reset  
False = channel pulse counting\*

\*Provided channel is not currently in a hardware reset.

### Requirements

None

### Specifications

Syntax	Data Type	Range	Access
S<slot>:C<channel>.RES	Boolean	1-2	Write Only

### Examples

Address	Value	Description
---------	-------	-------------

S1:C1.RES	0	Slot 1, ch1 not software reset.
S1:C2.RES	1	Slot 1, ch2 software reset.

### Channel Enable (Hardware) Addressing

Channels can be enabled by an external enable signal. Enable/disable this capability by referencing address type CCFG<channel>.HEN.

#### Values

True = allow hardware enable of channel  
False = don't allow hardware enable of channel

#### Requirements

None

#### Specifications

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.HEN	Boolean	1-2	Write Only

#### Examples

Address	Value	Description
S1:CCFG1.HEN	0	Slot 1, ch1 hardware enable not allowed.
S1:CCFG2.HEN	1	Slot 1, ch2 hardware enabled allowed.

### Channel Reset (Hardware) Addressing

A channel's pulse count can be reset by an external reset signal. Enable/disable this capability by referencing address type CCFG<channel>.HRES.

#### Values

True = allow hardware reset of pulse count  
False = don't allow hardware reset of pulse count

#### Requirements

None

#### Specifications

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.HRES	Boolean	1-2	Write Only

#### Examples

Address	Value	Description
S1:CCFG1.HRES	0	Slot 1, ch1 hardware reset not allowed.
S1:CCFG2.HRES	1	Slot 1, ch2 hardware reset allowed.

### Channel Debounce Count Addressing

CCFG<channel>.DBNC establishes the maximum pulse frequency that a channel will count.

#### Values

2 = 15 kHz  
4 = 10 kHz  
8 = 5 kHz  
16 = 2.5 kHz  
40 = 1 kHz

**Note:** The default value is 15 kHz.

#### Requirements

None

#### Specifications

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.DBNC	Byte, Word, Short, DWord, Long	1-2	Write Only

**Examples**

Address	Value	Description
S1:CCFG1.DBNC	40	Slot 1, ch1 max pulse freq set to 1 kHz.
S1:CCFG2.DBNC	2	Slot 1, ch2 max pulse freq set to 15 kHz.

**8 Digital Input Addressing**

The counter's general-purpose inputs are referenced using address type CDI.

**Requirements**

None

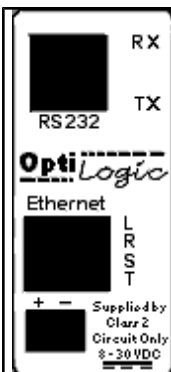
**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CDI<point>	<b>Boolean</b>	0-7	Read Only
S<slot>:CDI<offset>*	Byte, Word, Short, DWord, Long	0	Read Only

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

**Examples**

Address	Value	Description
S1:CDI0	1	Slot 1, point 0 (terminal 10) is on.
S1:CDI0 (as Byte)	0	Slot 1, byte 0 (points 0-7 are all off).

**Base RS232 Port**

The Ethernet Base contains an RS232 serial port. Both the transmit buffer and receive buffer of the driver are 48 bytes in size. Likewise, the corresponding tags can be a maximum of 48 bytes. Incoming bytes are appended to the receive buffer as long as they are received in proper time. This time period depends on the baud rate and is based on a 20-character delay using a 20 ms resolution.

300 Baud => 660 ms

1200 Baud => 160 ms

2400 Baud => 80 ms

4800 Baud => 40 ms

9600 Baud => 20 ms

19200 Baud => 20 ms

For a 4800 baud link, bytes would be appended to the receive buffer as long as they were received within 40 ms of the last byte sent. After 40 ms, any incoming bytes are

treated as a new stream. The receive buffer would clear and only contain these new bytes.

If the receive buffer is full and additional bytes are received within the proper time frame, the buffer will reset with these additional bytes. The first 48 bytes will be lost.

Below is a list of possible configurations:

1. Baud rates: 300, 1200, 2400, 4800, 9600 and 19200.
2. Data bits: 7 or 8
3. Parity: none, odd or even
4. Stop bits: 1, 1.5 or 2

An RJ45 connector is required.

### Subtypes

OL4054, OL4058

### Address Types

[Serial Input: Data](#)

[Serial Input: Number of Received Bytes](#)

[Serial Input: Parity Error](#)

[Serial Output: Data](#)

[Serial Output: Number of Bytes Sent](#)

[Serial Port Configuration: Baud Rate](#)

[Serial Port Configuration: #Data Bits](#)

[Serial Port Configuration: Parity](#)

[Serial Port Configuration: #Stop Bits](#)

[Serial Port Configuration: Set](#)

### Serial Input: Data Addressing

To receive serial data, reference address type SI<port>.DATA.

**Note:** The default configuration parameters are 300, n, 8 and 1.

### Specifications

Syntax	Data Type	Range	Access
B:SI<port>.DATA [r][c]*	Byte Array, Char Array	0	Read Only
B:SI<port>.DATA	String	0	Read Only

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

### Examples

Address	Value	Description
B:SI0.DATA	"hello"	port 0 input data viewed as a string
B:SI0.DATA [2][2]	[105, 105][105, 105]	port 0 input data in array form. In string form this would equate to "iiii"

### Serial Input: Number of Received Bytes Addressing

The number of received serial bytes (number of bytes in SI<port>.DATA. can be accessed by referencing address type SI<port>.NUMBYTES. If bytes are received within the timeout period mentioned above and are therefore appended to the input DATA buffer, NUMBYTES will reflect the total number of bytes in the input DATA buffer and not the number of bytes received on an individual block read. NUMBYTES will reset upon receiving a new stream.

### Requirements

None

### Specifications

Syntax	Data Type	Range	Access
B:SI<port>:NUMBYTES	Byte, Word, Short, DWord, Long	0	Read Only

### Examples

Address	Value	Description
B:SI0.NUMBYTES	0	port 0 input, no bytes in input DATA buffer
B:SI0.NUMBYTES	5	port 0 input, 5 bytes in input DATA buffer

### Serial Input: Parity Error Addressing

Reference SI<port>.PARITYERR to determine whether a parity error occurred on the last block read of the serial input port.

### Values

True = Parity error occurred  
False = No parity error occurred

### Requirements

None

### Specifications

Syntax	Data Type	Range	Access
B:SI<port>.PARITYERR	Boolean	0	Read Only

### Examples

Address	Value	Description
B:SI0.PARITYERR	0	port 0 input, no error
B:SI0.PARITYERR	1	port 0 input, parity error occurred

### Serial Output: Data Addressing

To transmit serial data, reference address type SO<port>.DATA.

**Note:** The default configuration parameters are 300, n, 8 and 1.

### Specifications

Syntax	Data Type	Range	Access
B:SO<port>.DATA [r][c]*	Byte Array, Char Array	0	Write Only
B:SO<port>.DATA	String	0	Write Only

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

### Examples

Address	Value	Description
B:SO0.DATA	"hello"	port 0 output, transmit "hello"
B:SO0.DATA [2][2]	[105, 105][105, 105]	port 0 output data in array form. In string form this would equate to transmitting "iiii"

### Serial Output: Number of Bytes Sent

Reference SO<port>.BYTESENT to determine how many bytes were sent on the last transmission. This value is available after transmission of serial data.

### Requirements

None

### Specifications

Syntax	Data Type	Range	Access
B:SO<port>.BYTESENT	Byte, Word, Short, DWord, Long	0	Read Only

### Examples



Address	Value	Description
B:SO0.BYTESSENT	0	port 0 output, no bytes sent on last transmission
B:SO0.BYTESSENT	47	port 0 output, 47 bytes sent

### Serial Port Configuration: Baud Rate

To configure the baud rate for a serial port, reference SCFG<port>.BAUD

#### Values

1 = 300  
 2 = 1200  
 3 = 2400  
 4 = 4800  
 5 = 9600  
 6 = 19200

**Note:** The default value is 300 baud.

#### Requirements

None

#### Specifications

Syntax	Data Type	Range	Access
B:SCFG<port>.BAUD	Byte, Word, Short, DWord, Long	0	Write Only

#### Examples

Address	Value	Description
B:SCFG0.BAUD	4	port 0, baud = 4800
B:SCFG0.BAUD	6	port 0, baud = 19200

### Serial Port Configuration: #Data Bits

To configure the number of data bits for a serial port, reference SCFG<port>.DATABITS

#### Values

7 or 8

**Note:** The default value is 8 data bits.

#### Requirements

None

#### Specifications

Syntax	Data Type	Range	Access
B:SCFG<port>.DATABITS	Byte, Word, Short, DWord, Long	0	Write Only

#### Examples

Address	Value	Description
B:SCFG0.DATABITS	7	port 0 configured for 7 data bits
B:SCFG0.DATABITS	8	port 0 configured for 8 data bits

### Serial Port Configuration: Parity

To configure the parity for a serial port, reference SCFG<port>.BAUD

#### Values

0 = none  
 1 = odd  
 2 = even

**Note:** The default value is none.

#### Requirements:

None

**Specifications**

Syntax	Data Type	Range	Access
B:SCFG<port>.PARITY	Byte, Word, Short, DWord, Long	0	Write Only

**Examples**

Address	Value	Description
B:SCFG0.PARITY	0	port 0 configured for no parity
B:SCFG0.PARITY	2	port 0 configured for even parity

**Serial Port Configuration: #Stop Bits**

To configure the number of stop bits for a serial port, reference SCFG<port>.STOPBITS

**Values**

1 = 1  
2 = 2  
3 = 1.5

**Note:** The default value is 1.

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
B:SCFG<port>.STOPBITS	Byte, Word, Short, DWord, Long	0	Write Only

**Examples**

Address	Value	Description
B:SCFG0.STOPBITS	1	port 0 configured for 1 stop bit
B:SCFG0.STOPBITS	3	port 0 configured for 1.5 stop bits

**Serial Port Configuration: Set**

For any of the serial port configuration parameters (baud, parity and so forth) to be sent to the device, SCFG.SET must be set. Immediately after the parameters are sent, SCFG.SET will be reset.

**Values**

True = Send serial port configurations to device  
False = No action

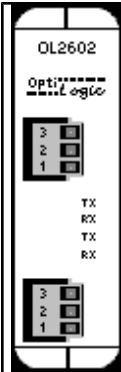
**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
B:SCFG<port>.SET	Boolean	N/A	Write Only

**Dual RS232 Port Module**



The OL2602 is a dual RS232 port serial module. Both the transmit buffer and receive buffer of the driver are 48 bytes in size. Likewise, the corresponding tags can be a maximum of 48 bytes. Incoming bytes are appended to the receive buffer as long as they are received in proper time. This time period depends on the baud rate and is based on a 20-character delay using a 20 ms resolution.

1200 Baud => 160 ms

2400 Baud => 80 ms

4800 Baud => 40 ms

9600 Baud => 20 ms

19200 Baud => 20 ms

For a 4800 baud link, bytes would be appended to the receive buffer as long as they were received within 40 ms of the last byte sent. After 40 ms, any incoming bytes are treated as a new stream. The receive buffer would clear and only contain these new bytes.

If the receive buffer is full and additional bytes are received within the proper time frame, the buffer will reset with these additional bytes. The first 48 bytes will be lost.

Below is a list of possible configurations:

1. Baud rates: 1200, 2400, 4800, 9600 and 19200.
2. Data bits: 7 or 8
3. Parity: none, odd or even
4. Stop bits: 1, 1.5 or 2

If using the OL4058 RTU, a maximum of one OL2602 modules may be used and it must be placed in slot 0. For the OL4054 RTU, two OL2602 may be used and they must be in either slot 0 or 1.

Port 1 is the top terminal strip, Port 2 is the bottom terminal strip. Below is the pin assignments:

- 1: Signal ground
- 2: TX
- 3: RX

### Subtypes

OL2602

### Address Types

[Serial Input: Data](#)

[Serial Input: Number of Received Bytes](#)[Serial Input: Parity Error](#)[Serial Output: Data](#)[Serial Output: Number of Bytes Sent](#)[Serial Port Configuration: Baud Rate](#)[Serial Port Configuration: #Data Bits](#)[Serial Port Configuration: Parity](#)[Serial Port Configuration: #Stop Bits](#)[Serial Port Configuration: Set](#)**Serial Input: Data Addressing**

To receive serial data, reference address type SI<port>.DATA.

**Note:** The default configuration parameters are 9600, n, 8 and 1.

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:SI<port>.DATA [r][c]*	Byte Array, Char Array	1-2	Read Only
S<slot>:SI<port>.DATA	String	1-2	Read Only

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

**Examples**

Address	Value	Description
S0:SI1.DATA	"hello"	Slot 0, port 1 input data viewed as a string.
S0:SI2.DATA [2][2]	[105, 105] [105, 105]	Slot 0, port 2 input data in array form. In string form this would equate to "iiii".

**Serial Input: Number of Received Bytes Addressing**

The number of received serial bytes (number of bytes in SI<port>.DATA) can be accessed by referencing address type SI<port>.NUMBYTES. If bytes are received within the timeout period mentioned above and are therefore appended to the input DATA buffer, NUMBYTES will reflect the total number of bytes in the input DATA buffer and not the number of bytes received on an individual block read. NUMBYTES will reset upon receiving a new stream.

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:SI<port>.NUMBYTES	Byte, Word, Short, DWord, Long	1-2	Read Only

**Examples**

Address	Value	Description
S1:SI1.NUMBYTES	0	Slot 1, port 1 input, no bytes in input DATA buffer
S1:SI2.NUMBYTES	5	Slot1, port2 input, 5 bytes in input DATA buffer

**Serial Input: Parity Error Addressing**

Reference SI<port>.PARITYERR to determine whether a parity error occurred on the last block read of the serial input port.

**Values**

True = Parity error occurred

False = No parity error occurred

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:SI<port>.PARITYERR	Boolean	1-2	Read Only

### Examples

Address	Value	Description
S1:SI1.PARITYERR	0	Slot 1, port 1 input, no error.
S1:SI2.PARITYERR	1	Slot 1, port 2 input, parity error occurred.

### Serial Output: Data Addressing

To transmit serial data, reference address type SO<port>.DATA.

**Note:** The default configuration parameters are 9600, n, 8 and 1.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:SO<port>.BYTESENT	Byte, Word, Short, DWord, Long	1-2	Read Only

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

### Examples

Address	Value	Description
S0:SO1.DATA	"hello"	Slot 0, port 1 output, transmit "hello".
S0:SO2.DATA [2][2]	[105, 105][105, 105]	Slot 0, port 2 output data in array form. In string form this would equate to transmitting "iiii".

### Serial Output: Number of Bytes Sent

Reference SO<port>.BYTESENT to determine how many bytes were sent on the last transmission. This value is available after transmission of serial data.

### Requirements

None

### Specifications

Syntax	Data Type	Range	Access
S<slot>:SO<port>.BYTESENT	Byte, Word, Short, DWord, Long	1-2	Read Only

### Examples

Address	Value	Description
S1:SO1.BYTESENT	0	Slot 1, port 1 output, no bytes sent on last transmission.
S1:SO1.BYTESENT	47	Slot 1, port 1 output, 47 bytes sent.

### Serial Port Configuration: Baud Rate

To configure the baud rate for a serial port, reference SCFG<port>.BAUD

### Values

2 = 1200  
 3 = 2400  
 4 = 4800  
 5 = 9600  
 6 = 19200

**Note:** The default value is 9600 baud.

### Requirements

None

### Specifications

Syntax	Data Type	Range	Access
S<slot>:SCFG<port>.BAUD	Byte, Word, Short, DWord, Long	1-2	Write Only

**Examples**

Address	Value	Description
S1:SCFG1.BAUD	4	Slot 1, port 1, baud = 4800.
S1:SCFG2.BAUD	6	Slot 1, port 2, baud = 19200.

**Serial Port Configuration: #Data Bits**

To configure the number of data bits for a serial port, reference SCFG<port>.DATABITS

**Values**

7 or 8

**Note:** The default value is 8 data bits.

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:SCFG<port>.DATABITS	Byte, Word, Short, DWord, Long	1-2	Write Only

**Examples**

Address	Value	Description
S1:SCFG1.DATABITS	7	Slot 1, port 1 configured for 7 data bits.
S1:SCFG2.DATABITS	8	Slot 1, port 2 configured for 8 data bits.

**Serial Port Configuration: Parity**

To configure the parity for a serial port, reference SCFG<port>.BAUD

**Values**

0 = none  
1 = odd  
2 = even

**Note:** The default value is none.

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:SCFG<port>.PARITY	Byte, Word, Short, DWord, Long	1-2	Write Only

**Examples**

Address	Value	Description
S1:SCFG1.PARITY	0	Slot 1, port 1 configured for no parity.
S1:SCFG2.PARITY	2	Slot 1, port 2 configured for even parity.

**Serial Port Configuration: #Stop Bits**

To configure the number of stop bits for a serial port, reference SCFG<port>.STOPBITS

**Values**

1 = 1  
2 = 2  
3 = 1.5

**Note:** The default value is 1.

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:SCFG<port>.STOPBITS	Byte, Word, Short, DWord, Long	1-2	Write Only

### Examples

Address	Value	Description
S1:SCFG1.STOPBITS	1	Slot 1, port 1 configured for 1 stop bit.
S1:SCFG2.STOPBITS	3	Slot 1, port 2 configured for 1.5 stop bits.

### Serial Port Configuration: Set

For any of the serial port configuration parameters (baud, parity and so forth) to be sent to the device, SCFG.SET must be set. Immediately after the parameters are sent, SCFG.SET will be reset.

### Values

True = Send serial port configurations to device  
False = No action

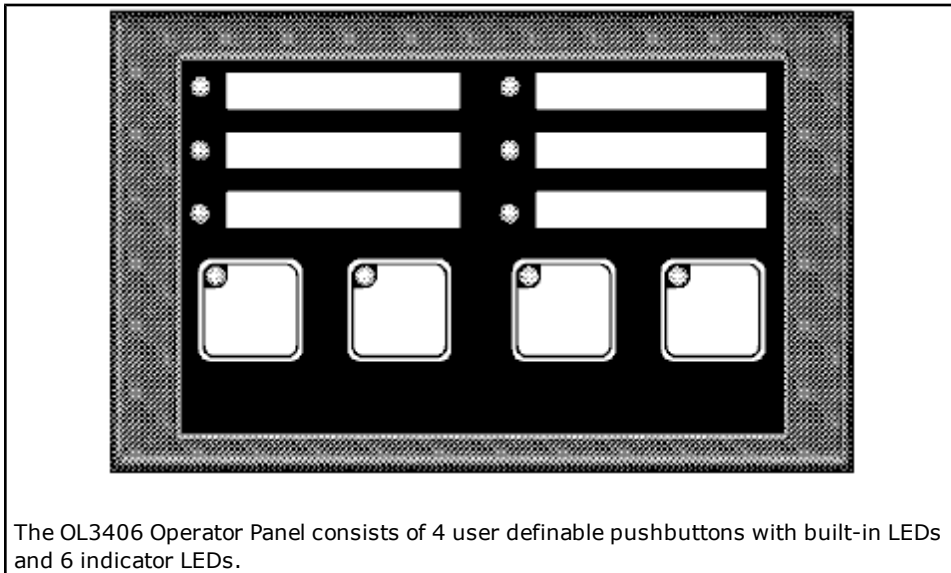
### Requirements

None

### Specifications

Syntax	Data Type	Range	Access
S<slot>:SCFG<port>.SET	Boolean	N/A	Write Only

## OL3406 Operator Panel



The OL3406 Operator Panel consists of 4 user definable pushbuttons with built-in LEDs and 6 indicator LEDs.

### Subtypes

OL3406

### Address Types

[Pushbutton LED On State](#)  
[Pushbutton LED Flash State](#)  
[Pushbutton LED Separation State](#)  
[Pushbutton State](#)  
[Pushbutton Configuration](#)  
[Force Pushbutton State](#)  
[Indicator LED On State](#)  
[Indicator LED Flash State](#)

### Pushbutton LED On State Addressing

Each button LED can be forced On/Off without the button being pressed. This can be achieved by referencing address type BTNLED<button>.ON.

### Values

True = on  
False = off

### Requirements

Button LED Separation state must be set. Button(s) must also be configured for momentary action. For more information, refer to [PushButton Configuration](#).

### Specifications

Syntax	Data Type	Range	Access
P:BTNLED<button number>.ON	Boolean	0-3	Write Only

### Examples

Address	Value	Description
P:BTNLED0.ON	1	Turn button 0 LED on (left most button).*
P:BTNLED3.ON	1	Turn button 3 LED on (right most button).*

\*See Requirements above.

### Pushbutton LED Flash State Addressing

Each button LED can be forced to flash On/Off without the button being pressed. This can be achieved by referencing address type BTNLED<button>.FLASH.

### Values

True = flash on  
False = flash off

### Requirements

Button LED ON state must be set for the corresponding button and button LED Separation state must also be set. Button(s) must be configured for momentary action. For more information, refer to [PushButton Configuration Addressing](#).

### Specifications

Syntax	Data Type	Range	Access
P:BTNLED<button number>.FLASH	Boolean	0-3	Write Only

### Examples

Address	Value	Description
P:BTNLED0.FLASH	1	Flash button 0 LED (left most button).*
P:BTNLED3.FLASH	1	Flash button 3 LED (right most button).*

\*See Requirements above.

### Pushbutton LED Separation State Addressing

When LED Separation is set, one is capable of controlling the on and flash state of individual button LEDs (see Pushbutton LED On and Flash State Addressing above).

### Values

True = separation on  
False = separation off

### Requirements

None.

### Specifications

Syntax	Data Type	Range	Access
P:LEDSEP	Boolean	N/A	Read/Write

### Pushbutton State Addressing



Button state (pressed/not pressed) can be monitored by referencing address type BTNSTATUS<button>.

### Values

True = pressed  
False = not pressed

### Requirements

None.

**Note:** Button state depends on the button configuration. For more information, refer to [Pushbutton Configuration Addressing](#).

### Specifications

Syntax	Data Type	Range	Access
P:BTNSTATUS<button number>	Boolean	0-3	Read Only

### Examples

Address	Value	Description
P:BTNSTATUS0	1	Button 0 is pressed (left most button).*
P:BTNSTATUS3	0	Button 3 is not pressed (right most button).

\*See Note above.

### Pushbutton Configuration Addressing

Each button can be configured to either latch their state (alternate action) or hold it momentarily while its being pressed/not pressed.

### Values

True = alternate action  
False = momentary action

### Requirements

None.

### Specifications

Syntax	Data Type	Range	Access
P:BTNCFG<button number>	Boolean	0-3	Read/Write

### Examples

Address	Value	Description
P:BTNCFG0	1	Button 0 is configured for alternate action.
P:BTNCFG3	0	Button 3 is configured for momentary action.

\*See Note above.

### Force Pushbutton State Addressing

Each button can be forced to a desired state. There are three means of forcing button state.

EQUALS:	Desired button state = specified state
OR:	Desired button state = (current state BITWISE OR specified state)
AND:	Desired button state = NOT(current state BITWISE AND specified state)

### Values (Desired button states)

True = button on ("pressed")  
False = button off ("not pressed")

### Requirements

Button must be configured for alternate action.

### Specifications

Below are the three means of forcing button state in detail:

Syntax	Data Type	Range	Access
P:BTNFORCE<button number>.EQUALS	Boolean	0-3	Write Only
P:BTNFORCE<button number>.OR	Boolean	0-3	Write Only
P:BTNFORCE<button number>.AND	Boolean	0-3	Write Only

### Examples

Address	Value	Description
P:BTNFORCE0.EQUALS	1	Force button 0 state to be on.
P:BTNFORCE0.OR	1	OR current state with 1. Force button 0 state to be on.
P:BTNFORCE0.AND	1	If button 0 state is currently on, set state to off. Otherwise, do nothing.

\*See Requirements above.

### Indicator LED On State Addressing

Each indicator LED can be forced On/Off by referencing address type INDLED<led>.ON.

#### Values

True = on  
False = off

#### Requirements

None.

#### Specifications

Syntax	Data Type	Range	Access
P:INDLED<led number>.ON	Boolean	0-5	Write Only

### Examples

Address	Value	Description
P:INDLED0.ON	1	Turn indicator LED 0 on (upper left LED).
P:INDLED5.ON	1	Turn indicator LED 5 on (bottom right LED).

### Indicator LED Flash State Addressing

Each indicator LED can be forced to flash On/Off by referencing address type INDLED<led>.FLASH.

#### Values

True = flash on  
False = flash off

#### Requirements

Indicator LED ON state must be set for the corresponding LED.

#### Specifications

Syntax	Data Type	Range	Access
P:INDLED<led number>.FLASH	Boolean	0-5	Write Only

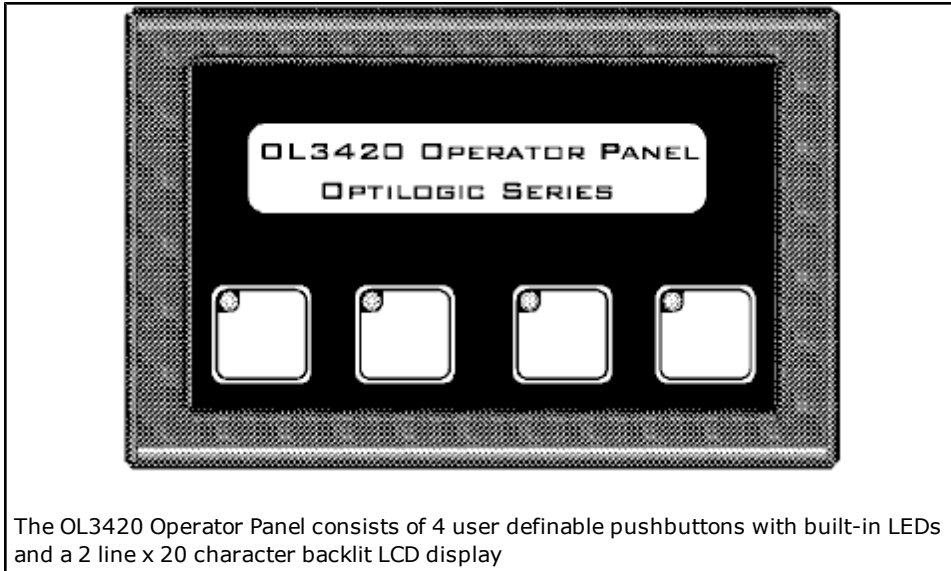
### Examples

Address	Value	Description
P:INDLED1.FLASH	1	Flash indicator LED 1 (upper right LED).*
P:INDLED4.FLASH	1	Flash indicator LED 4 (lower left LED).*

\*See Requirements above.

## OL3420 Operator Panel

---



### Subtypes

OL3420

### Address Types

[Pushbutton LED On State](#)  
[Pushbutton LED Flash State](#)  
[Pushbutton LED Separation State](#)  
[Pushbutton State](#)  
[Pushbutton Configuration](#)  
[Force Pushbutton State](#)  
[Alphanumeric Display](#)

### Pushbutton LED On State Addressing

Each button LED can be forced On/Off without the button being pressed. This can be achieved by referencing address type `BTNLED<button>.ON`.

### Values

True = on  
False = off

### Requirements

Button LED Separation state must be set. Button(s) must also be configured for momentary action. For more information, refer to [PushButton Configuration](#).

### Specifications

Syntax	Data Type	Range	Access
P:BTNLED<button number>.ON	Boolean	0-3	Write Only

### Examples

Address	Value	Description
P:BTNLED0.ON	1	Turn button 0 LED on (left most button).*
P:BTNLED3.ON	1	Turn button 3 LED on (right most button).*

\*See Requirements above.

### Pushbutton LED Flash State Addressing

Each button LED can be forced to flash On/Off without the button being pressed. This can be achieved by referencing address type `BTNLED<button>.FLASH`.

### Values

True = flash on  
False = flash off

**Requirements**

Button LED ON state must be set for the corresponding button and button LED Separation state must be set. Button(s) must also be configured for momentary action. For more information, refer to [PushButton Configuration](#).

**Specifications**

Syntax	Data Type	Range	Access
P:BTNLED<button number>.FLASH	Boolean	0-3	Write Only

**Examples**

Address	Value	Description
P:BTNLED0.FLASH	1	Flash button 0 LED (left most button).*
P:BTNLED3.FLASH	1	Flash button 3 LED (right most button).*

\*See Requirements above.

**Pushbutton LED Separation State Addressing**

When LED Separation is set, one is capable of controlling the on and flash state of individual button LEDs (see Pushbutton LED On and Flash State Addressing above).

**Values**

True = separation on  
False = separation off

**Requirements**

None.

Syntax	Data Type	Range	Access
P:LEDSEP	Boolean	N/A	Read/Write

**Pushbutton State Addressing**

Button state (pressed/not pressed) can be monitored by referencing address type BTNSTATUS<button>.

**Values**

True = pressed  
False = not pressed

**Requirements**

None.

**Note:** Button state depends on the button configuration. See [Pushbutton Configuration Addressing](#) below.

**Specifications**

Syntax	Data Type	Range	Access
P:BTNSTATUS<button number>	Boolean	0-3	Read Only

**Examples**

Address	Value	Description
P:BTNSTATUS0	1	Button 0 is pressed (left most button).*
P:BTNSTATUS3	0	Button 3 is not pressed (right most button).

\*See Note above.

**Pushbutton Configuration Addressing**

Each button can be configured to either latch their state (alternate action) or hold it momentarily while its being pressed/not pressed.

**Values**

True = alternate action  
False = momentary action

**Requirements**

None.

**Specifications**

Syntax	Data Type	Range	Access
P:BTNCFG<button number>	Boolean	0-3	Read/Write

**Examples**

Address	Value	Description
P:BTNCFG0	1	Button 0 is configured for alternate action.
P:BTNCFG3	0	Button 3 is configured for momentary action.

\*See Note above.

**Force Pushbutton State Addressing**

Each button can be forced to a desired state. There are three ways to force button state.

EQUALS:	Desired button state = specified state
OR:	Desired button state = (current state BITWISE OR specified state)
AND:	Desired button state = NOT(current state BITWISE AND specified state)

**Values (Desired button states)**

True = button on ("pressed")

False = button off ("not pressed")

**Requirements**

Button must be configured for alternate action.

**Specifications**

Below are the three means of forcing button state in detail.

Syntax	Data Type	Range	Access
P:BTNFORCE<button number>.EQUALS	Boolean	0-3	Write Only
P:BTNFORCE<button number>.OR	Boolean	0-3	Write Only
P:BTNFORCE<button number>.AND	Boolean	0-3	Write Only

**Examples**

Address	Value	Description
P:BTNFORCE0.EQUALS	1	Force button 0 state to be on.
P:BTNFORCE0.OR	1	OR current state with 1. Force button 0 state to be on.
P:BTNFORCE0.AND	1	If button 0 state is currently on, set state to off. Otherwise, do nothing.

\*See Requirements above.

**Alphanumeric Display Addressing**

There are two lines of display for alphanumeric strings of length 20 characters or less.

**Requirements**

None.

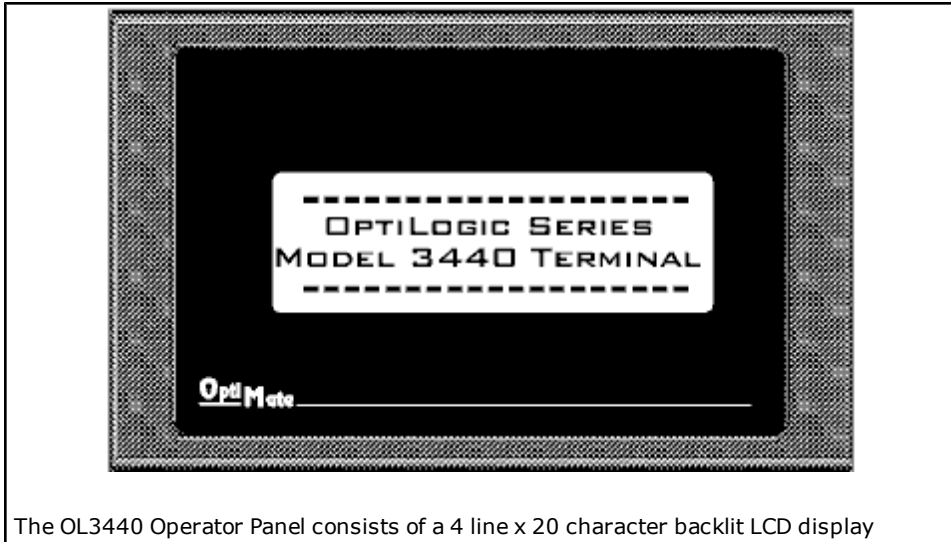
**Specifications**

Syntax	Data Type	Range	Access
P:LINE<line number>	String	0-1	Write Only

**Examples**

Syntax	Data Type	Range	Access
P:LINE<line number>	String	0-1	Write Only

## OL3440 Operator Panel



### Subtypes

OL3440

### Alphanumeric Display Addressing

There are four lines of display for alphanumeric strings of length 20 characters or less.

### Requirements

None.

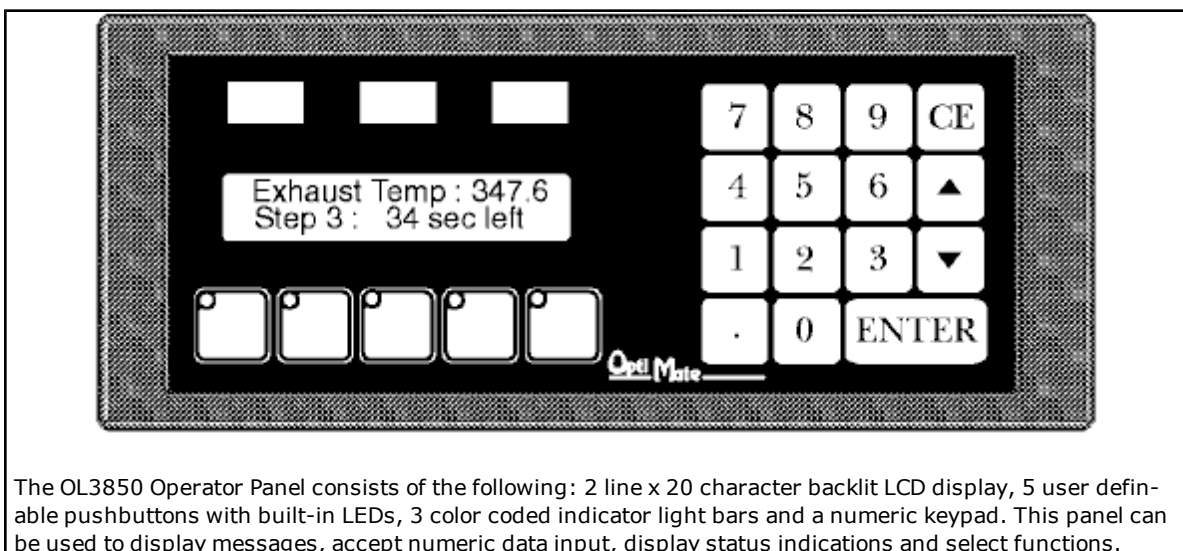
### Specifications

Syntax	Data Type	Range	Access
P:LINE<line number>	String	0-3	Write Only

### Examples

Address	Value	Description
P:LINE0	"hello"	Set line 0 (top line) string to "hello".
P:LINE3	"world"	Set line 3 (bottom line) string to "world".

## OL3850 Operator Panel



**Subtypes**

OL3850

**Address Types**

[Pushbutton LED On State](#)  
[Pushbutton LED Flash State](#)  
[Pushbutton LED Separation State](#)  
[Pushbutton State](#)  
[Pushbutton Configuration](#)  
[Force Pushbutton State](#)  
[Light Bar On State](#)  
[Light Bar Flash State](#)  
[Alphanumeric Display](#)  
[Keypad Data](#)  
[Keypad Data Available](#)  
[Keypad Arrow Max](#)  
[Keypad Arrow Min](#)

**Pushbutton LED On State Addressing**

Each button LED can be forced On/Off without the button being pressed. This can be achieved by referencing address type `BTNLED<button>.ON`.

**Values**

True = on  
False = off

**Requirements**

Button LED Separation state must be set. Button(s) must also be configured for momentary action. For more information, refer to [PushButton Configuration Addressing](#).

**Specifications**

Syntax	Data Type	Range	Access
P:BTNLED<button number>.ON	Boolean	0-4	Write Only

**Examples**

Address	Value	Description
P:BTNLED0.ON	1	Turn button 0 LED on (left most button).*
P:BTNLED4.ON	1	Turn button 4 LED on (right most button).*

\*See Requirements above.

**Pushbutton LED Flash State Addressing**

Each button LED can be forced to flash On/Off without the button being pressed. This can be achieved by referencing address type `BTNLED<button>.FLASH`.

**Values**

True = flash on  
False = flash off

**Requirements**

Button LED ON state must be set for the corresponding button and button LED Separation state must be set. Button(s) must also be configured for momentary action. For more information, refer to [PushButton Configuration Addressing](#).

Syntax	Data Type	Range	Access
P:BTNLED<button number>.FLASH	Boolean	0-4	Write Only

**Examples**

Address	Value	Description
P:BTNLED0.FLASH	1	Flash button 0 LED (left most button).*

P:BTNLED4.FLASH	1	Flash button 4 LED (right most button).*
-----------------	---	--

\*See Requirements above.

### Pushbutton LED Separation State Addressing

When LED Separation is set, one is capable of controlling the on and flash state of individual button LEDs. For more information, refer to "Pushbutton LED On and Flash State Addressing" above.

#### Values

True = separation on  
False = separation off

#### Requirements

None.

#### Specifications

Syntax	Data Type	Range	Access
P:LEDSEP	Boolean	N/A	Read/Write

### Pushbutton State Addressing

Button state (pressed/not pressed) can be monitored by referencing address type BTNSTATUS<button>.

#### Values

True = pressed  
False = not pressed

#### Requirements

None.

**Note:** Button state depends on the button configuration. For more information, refer to [Pushbutton Configuration Addressing](#) below.

#### Specifications

Syntax	Data Type	Range	Access
P:BTNSTATUS<button number>	Boolean	0-4	Read Only

#### Examples

Address	Value	Description
P:BTNSTATUS0	1	Button 0 is pressed (left most button).*
P:BTNSTATUS4	0	Button 4 is not pressed (right most button).

\*See Note above.

### Pushbutton Configuration Addressing

Each button can be configured to either latch their state (alternate action) or hold it momentarily while its being pressed/not pressed.

#### Values

True = alternate action  
False = momentary action

#### Requirements

None.

#### Specifications

Syntax	Data Type	Range	Access
P:BTNCFG<button number>	Boolean	0-4	Read/Write

#### Examples

Address	Value	Description
P:BTNCFG0	1	Button 0 is configured for alternate action.



P:BTNCFG4	0	Button 4 is configured for momentary action.
-----------	---	--

### Force Pushbutton State Addressing

Each button can be forced to a desired state. There are three ways to force button state.

EQUALS:	Desired button state = specified state
OR:	Desired button state = (current state BITWISE OR specified state)
AND:	Desired button state = NOT(current state BITWISE AND specified state)

### Values (Desired button states)

True = button on ("pressed")

False = button off ("not pressed")

### Requirements

Button must be configured for alternate action.

### Specifications

Below are the three means of forcing button state in detail:

Syntax	Data Type	Range	Access
P:BTNFORCE<button number>.EQUALS	Boolean	0-4	Write Only
P:BTNFORCE<button number>.OR	Boolean	0-4	Write Only
P:BTNFORCE<button number>.AND	Boolean	0-4	Write Only

### Examples

Address	Value	Description
P:BTNFORCE0.EQUALS	1	Force button 0 state to be on.*
P:BTNFORCE0.OR	1	OR current state with 1. Force button 0 state to be on.*
P:BTNFORCE0.AND	1	If button 0 state is currently on, set state to off. otherwise, do nothing.*

\*See Requirements above.

### Light Bar On State Addressing

Each light bar can be forced On/Off by referencing address type LITEBAR<bar>.ON.

### Values

True = on

False = off

### Requirements

None.

### Specifications

Syntax	Data Type	Range	Access
P:LITEBAR<light bar>.ON	Boolean	0-2	Write Only

### Examples

Address	Value	Description
P:LITEBAR0.ON	1	Turn light bar 0 on (left most bar).
P:LITEBAR2.ON	1	Turn light bar 2 on (right most bar).

### Light Bar Flash State Addressing

Each light bar can be forced to flash On/Off by referencing address type LITEBAR<bar>.FLASH.

### Values

True = flash on

False = flash off

### Requirements

Light Bar ON state must be set for the corresponding light bar.

**Specifications**

Syntax	Data Type	Range	Access
P:LITEBAR<light bar>.FLASH	Boolean	0-2	Write Only

**Examples**

Address	Value	Description
P:LITEBAR0.FLASH	1	Flash light bar 0 LED (left most bar).*
P:LITEBAR2.FLASH	1	Flash light bar 2 LED (right most bar).*

\*See Requirements above.

**Alphanumeric Display Addressing**

There are two lines of display for alphanumeric strings of length 20 characters or less. Keypad data may also be inserted into the text string by using the caret (^) as a placeholder for each digit (including the decimal point) of the keypad data.

**Requirements**

None.

**Specifications**

Syntax	Data Type	Range	Access
P:LINE<line number>	String	0-1	Write Only

**Examples**

Address	Value	Description
P:LINE0	"hello"	Set line 0 (top line) string to "hello".
P:LINE1	"world"	Set line 1 (bottom line) string to "world".
P:LINE0	"^^^^^^"	If keypad data = 1234.56, then line 0 string will be "1234.56".*

\*For more information, refer to "Keypad Data Addressing" below.

**Keypad Data Addressing**

Numeric data can be read from (entered via keypad or this address) and written to (via this address) the panel keypad. Float precision can be specified by appending a bit (0 to 10 allowed) to the KDATA address. This will represent the floating point precision on any writes to the device. If no precision is specified (the default case), the precision will be set such that the number of digits in the integer and fractional parts sums up to 10. If a precision is specified such that the 10-digit limit is exceeded, the precision will be set to the default value previously discussed. Note that the value written to the device may differ from the value displayed in the client. This may be due to floating point round off and truncation errors from the driver, client or both. For more information on floating point precision, refer to [Device Setup](#).

**Note:** Keypad data may be altered using the arrows located on the keypad. Upper and lower bounds set by Arrow Max and Arrow Min respectively, will only limit the data set by the arrows, not the data set in KDATA. Only the data type can place constraints on the upper and lower limits of the keypad data when set using KDATA.

**Specifications**

Syntax	Data Type	Range	Access
P:KDATA	Double, Float, DWord	N/A	Read/Write
P:KDATA.<precision>	Double, Float, DWord	N/A	Read/Write

**Examples (Writes)**

Address	Value	Description
P.KDATA	10.123456	10.123456 will be written to the device.
P:KDATA.4 as a float	10.123456	10.1235 will be written to the device.
P:KDATA	9999999999	9999999999 will be written to the device.

**Keypad Data Available Addressing**

To determine if new keypad data has been entered at the panel, reference KDATAREADY. This flag can be cleared by writing to KDATA.

**Values**

True = new data has been entered  
False = all data has been read from driver, no new data

Syntax	Data Type	Range	Access
P:KDATAREADY	Boolean	N/A	Read Only

### Keypad Arrow Max Addressing

Panel keypad data entered via the arrows, can be upper bounded by referencing ARROW.MAX. Any keypad data entered above this max will automatically get set to this max value.

**Note:** Initially, before ARROW.MAX is set, the upper limit internally to the device is 999999999. When the limit is set, keypad data cannot exceed the size of ARROW.MAX which is 32 bits (DWord).

### Specifications

Syntax	Data Type	Range	Access
P:ARROW.MAX	DWord	N/A	Write Only

### Keypad Arrow Min Addressing

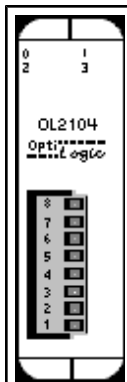
Panel keypad data entered via the arrows, can be lower bounded by referencing ARROW.MIN. Any keypad data entered below this min will automatically get set to this min value.

**Note:** Initially, before ARROW.MIN is set, the lower limit internally to the device is 0.

### Specifications

Syntax	Data Type	Range	Access
P:ARROW.MIN	DWord	N/A	Write Only

## 4 Digital Output Module



Digital outputs are used to turn loads on and off. Loads may be lights, motors, solenoids or any type of On/Off device.

Digital outputs in the OptiLogic series come in three types: relay, transistor and solid state relay. Each type has applications it is best suited for. For more information, refer to the module manual.

If there is a loss of communication with the host, the output module will enter a fail safe state. There are three types of states: fail safe to all outputs off, fail safe to a pattern and fail safe to last state. The time in which a fail safe state is entered after loss of communication can be delayed up to 25.5 seconds. If certain points need to be treated differently in fail safe mode, a pattern can be used to signify which points are turned on and which ones are turned off in fail safe mode.

### Subtypes

OL2104

### Address Types

[4 Digital Output](#)

[Fail Safe Type](#)

[Fail Safe Pattern](#)

[Fail Safe Time](#)

[Fail Safe Set](#)

### 4 Digital Output Addressing Specifications

Syntax	Data Type	Range	Access
S<slot>:DO<point>	Boolean	0-3	Read/Write
S<slot>:DO<offset>*	Byte**	0	Read/Write
S<slot>:DO<offset>*	Word, Short**	0	Read/Write
S<slot>:DO<offset>*	DWord, Long**	0	Read/Write

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

\*\*Since only the lower nibble of the least significant byte is being used, any value entered above 15 will be cropped.

### Examples

Address	Value	Description
S1:DO0	1	Slot 1, point 0 (turn point 0 on).
S1:DO0 (as Byte)	15	Slot 1, byte 0 (turn points 0-3 on).
S1:DO0 (as Word)	0	Slot 1, word 0 (turn all points off).
S1:DO0 (as DWORD)	15	Slot 1, DWord 0 (turn points 0-3 on).

### Fail Safe Type Addressing

There are three types of fail safe states: fail safe to all outputs off, fail safe to a pattern and fail safe to last state. These types may be specified by referencing FS.TYPE.

### Values

- 1 = Fail safe to all outputs off
- 2 = Fail safe to the pattern contained in FS<point>.PATTERN
- 3 = Fail safe to last state

### Requirements

None

**Important:** The value specified in FS.TYPE is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.TYPE	Byte, Word, Short, DWord, Long	N/A	Write Only

### Fail Safe Pattern Addressing

Using a fail safe pattern, users can specify which points are set and which ones are cleared when in fail safe mode.

### Values

- True = turn point on
- False = turn point off

### Requirements

FS.TYPE must be set to 2 for device to fail safe to a pattern.

**Note:** If using Byte, Word, Short, DWord or Long data types, the bit pattern determines which points are turned on and which ones are turned off.

**Important:** The pattern specified in FS.PATTERN is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>FS<point>.PATTERN	Boolean	0-3	Write Only
S<slot>:FS<offset>.PATTERN*	Byte**	0	Write Only
S<slot>:FS<offset>.PATTERN*	Word, Short**	0	Write Only
S<slot>:FS<offset>.PATTERN*	DWord, Long**	0	Write Only

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

\*\*Since only the lower nibble of the least significant byte is being used, any value entered above 15 will be cropped.

### Examples

Address	Value	Description
S1:FS0.PATTERN	1	Slot 1, turn point 0 on in fail safe mode.*
S1:FS0.PATTERN (as Byte)	15	Slot 1, turn points 0-3 on in fail safe mode.*

S1:FS0.PATTERN (as Word)	0	Slot 1, turn points 0-3 off in fail safe mode.*
S1:FS0.PATTERN (as DWORD)	15	Slot 1, turn points 0-3 on in fail safe mode.*

\*See Notes and Requirements above.

### Fail Safe Time Addressing

After communication between the host and the RTU has been lost, a time delay may be introduced before fail safe mode is engaged for the output module. This time delay can be entered by referencing FS.TIME and is specified in tenths of seconds.

### Requirements

None

**Important:** The value specified in FS.TYPE is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.TIME	Byte, Word, Short, DWord, Long	N/A	Write Only

### Examples

Address	Value	Description
S1:FS.TIME	255	Slot 1, fail safe time delay = 25.5 seconds.
S1:FS.TIME	0	Slot 1, no delay.
S1:FS.TIME	1	Slot 1, fail safe time delay = .1 seconds.

### Fail Safe Set Addressing

For any of the fail safe configuration parameters (type, time and pattern) to be sent to the device, FS.SET must be set. Immediately after the parameters are sent, FS.SET will be reset.

### Values

True = Send fail safe configurations to device  
False = No action

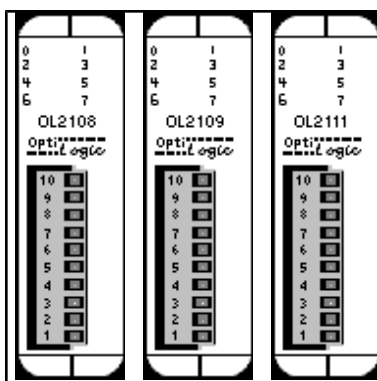
### Requirements

None

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.SET	Boolean	N/A	Write Only

## 8 Digital Output Module



Digital outputs are used to turn loads on and off. Loads may be lights, motors, solenoids or any type of On/Off device.

Digital outputs in the OptiLogic series come in three types: relay, transistor and solid state relay. Each type has applications it is best suited for. For more information, refer to the module manual.

If there is a loss of communication with the host, the output module will enter a fail safe state. There are three types of states: fail safe to all outputs off, fail safe to a pattern and fail safe to last state. The time in which a fail safe state is entered after loss of communication can be delayed up to 25.5 seconds. If certain points need to be treated differently in fail safe mode, a pattern can be used to signify which points are turned on and which ones are turned off in fail safe mode.

### Subtypes

OL2108, OL2109, OL2111

### Address Types

#### [8 Digital Output](#)

[Fail Safe Type](#)  
[Fail Safe Pattern](#)  
[Fail Safe Time](#)  
[Fail Safe Set](#)

## 8 Digital Output Addressing Specifications

Syntax	Data Type	Range	Access
S<slot>:DO<point>	Boolean	0-7	Read/Write
S<slot>:DO<offset>*	Byte	0	Read/Write
S<slot>:DO<offset>*	Word, Short	0	Read/Write
S<slot>:DO<offset>*	DWord, Long	0	Read/Write

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

### Examples

Address	Value	Description
S1:DO0	1	Slot 1, point 0 (turn point 0 on).
S1:DO0 (as Byte)	255	Slot 1, byte 0 (turn points 0-7 on).
S1:DO0 (as Word)	0	Slot 1, word 0 (turn points 0-7 off).
S1:DO0 (as DWORD)	255	Slot 1, DWord 0 (turn points 0-7 on).

### Fail Safe Type Addressing

There are three types of fail safe states: fail safe to all outputs off, fail safe to a pattern and fail safe to last state. These types may be specified by referencing FS.TYPE.

#### Values

- 1 = Fail safe to all outputs off
- 2 = Fail safe to the pattern contained in FS<point>.PATTERN
- 3 = Fail safe to last state

#### Requirements

None

**Important:** The value specified in FS.TYPE is not sent to the device until FS.SET is set.

#### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.TYPE	Byte, Word, Short, DWord, Long	N/A	Write Only

### Fail Safe Pattern Addressing

Using a fail safe pattern, users can specify which points are set and which ones are cleared when in fail safe mode.

#### Values

- True = turn point on
- False = turn point off

#### Requirements

FS.TYPE must be set to 2 for device to fail safe to a pattern.

**Note:** If using Byte, Word, Short, DWord or Long data types, the bit pattern determines which points are turned on and which ones are turned off.

**Important:** The pattern specified in FS.PATTERN is not sent to the device until FS.SET is set.

#### Specifications

Syntax	Data Type	Range	Access
S<slot>FS<point>.PATTERN	Boolean	0-7	Write Only
S<slot>:FS<offset>.PATTERN*	Byte	0	Write Only
S<slot>:FS<offset>.PATTERN*	Word, Short	0	Write Only
S<slot>:FS<offset>.PATTERN*	DWord, Long	0	Write Only

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

### Examples

Address	Value	Description
S1:FS0.PATTERN	1	Slot 1, turn point 0 on in fail safe mode.*
S1:FS0.PATTERN (as Byte)	255	Slot 1, turn points 0-7 on in fail safe mode.*
S1:FS0.PATTERN (as Word)	0	Slot 1, turn point 0-7 off in fail safe mode.*
S1:FS0.PATTERN (as DWORD)	255	Slot 1, turn points 0-7 on in fail safe mode.*

\*See Notes and Requirements above.

### Fail Safe Time Addressing

After communication between the host and the RTU has been lost, a time delay may be introduced before fail safe mode is engaged for the output module. This time delay can be entered by referencing FS.TIME and is specified in tenths of seconds.

### Requirements

None

**Important:** The value specified in FS.TYPE is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.TIME	Byte, Word, Short, DWord, Long	N/A	Write Only

### Examples

Address	Value	Description
S1:FS.TIME	255	Slot 1, fail safe time delay = 25.5 seconds.
S1:FS.TIME	0	Slot 1, no delay.
S1:FS.TIME	1	Slot 1, fail safe time delay = .1 seconds.

### Fail Safe Set Addressing

For any of the fail safe configuration parameters (type, time and pattern) to be sent to the device, FS.SET must be set. Immediately after the parameters are sent, FS.SET will be reset.

### Values

True = Send fail safe configurations to device

False = No action

### Requirements

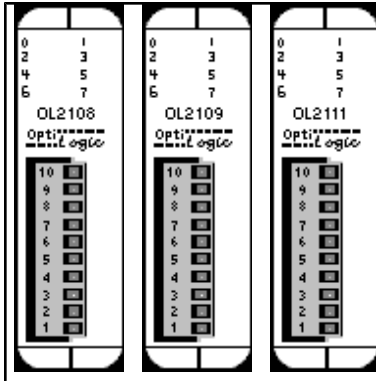
None

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.SET	Boolean	N/A	Write Only

## 8 Digital Output Module

---



Digital outputs are used to turn loads on and off. Loads may be lights, motors, solenoids or any type of On/Off device.

Digital outputs in the OptiLogic series come in three types: relay, transistor and solid state relay. Each type has applications it is best suited for. For more information, refer to the module manual.

If there is a loss of communication with the host, the output module will enter a fail safe state. There are three types of states: fail safe to all outputs off, fail safe to a pattern and fail safe to last state. The time in which a fail safe state is entered after loss of communication can be delayed up to 25.5 seconds. If certain points need to be treated differently in fail safe mode, a pattern can be used to signify which points are turned on and which ones are turned off in fail safe mode.

### Subtypes

OL2108, OL2109, OL2111

### Address Types

#### [8 Digital Output](#)

#### [Fail Safe Type](#)

#### [Fail Safe Pattern](#)

#### [Fail Safe Time](#)

#### [Fail Safe Set](#)

### 8 Digital Output Addressing Specifications

Syntax	Data Type	Range	Access
S<slot>:DO<point>	Boolean	0-7	Read/Write
S<slot>:DO<offset>*	Byte	0	Read/Write
S<slot>:DO<offset>*	Word, Short	0	Read/Write
S<slot>:DO<offset>*	DWord, Long	0	Read/Write

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

### Examples

Address	Value	Description
S1:DO0	1	Slot 1, point 0 (turn point 0 on).
S1:DO0 (as Byte)	255	Slot 1, byte 0 (turn points 0-7 on).
S1:DO0 (as Word)	0	Slot 1, word 0 (turn points 0-7 off).
S1:DO0 (as DWord)	255	Slot 1, DWord 0 (turn points 0-7 on).

### Fail Safe Type Addressing

There are three types of fail safe states: fail safe to all outputs off, fail safe to a pattern and fail safe to last state. These types may be specified by referencing FS.TYPE.

### Values

- 1 = Fail safe to all outputs off
- 2 = Fail safe to the pattern contained in FS<point>.PATTERN
- 3 = Fail safe to last state

### Requirements

None

**Important:** The value specified in FS.TYPE is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.TYPE	Byte, Word, Short, DWord, Long	N/A	Write Only

### Fail Safe Pattern Addressing



Using a fail safe pattern, users can specify which points are set and which ones are cleared when in fail safe mode.

### Values

True = turn point on  
False = turn point off

### Requirements

FS.TYPE must be set to 2 for device to fail safe to a pattern.

**Note:** If using Byte, Word, Short, DWord or Long data types, the bit pattern determines which points are turned on and which ones are turned off.

**Important:** The pattern specified in FS.PATTERN is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>FS<point>.PATTERN	Boolean	0-7	Write Only
S<slot>:FS<offset>.PATTERN*	Byte	0	Write Only
S<slot>:FS<offset>.PATTERN*	Word, Short	0	Write Only
S<slot>:FS<offset>.PATTERN*	DWord, Long	0	Write Only

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

### Examples

Address	Value	Description
S1:FS0.PATTERN	1	Slot 1, turn point 0 on in fail safe mode.*
S1:FS0.PATTERN (as Byte)	255	Slot 1, turn points 0-7 on in fail safe mode.*
S1:FS0.PATTERN (as Word)	0	Slot 1, turn point 0-7 off in fail safe mode.*
S1:FS0.PATTERN (as DWORD)	255	Slot 1, turn points 0-7 on in fail safe mode.*

\*See Notes and Requirements above.

### Fail Safe Time Addressing

After communication between the host and the RTU has been lost, a time delay may be introduced before fail safe mode is engaged for the output module. This time delay can be entered by referencing FS.TIME and is specified in tenths of seconds.

### Requirements

None

**Important:** The value specified in FS.TYPE is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.TIME	Byte, Word, Short, DWord, Long	N/A	Write Only

### Examples

Address	Value	Description
S1:FS.TIME	255	Slot 1, fail safe time delay = 25.5 seconds.
S1:FS.TIME	0	Slot 1, no delay.
S1:FS.TIME	1	Slot 1, fail safe time delay = .1 seconds.

### Fail Safe Set Addressing

For any of the fail safe configuration parameters (type, time and pattern) to be sent to the device, FS.SET must be set. Immediately after the parameters are sent, FS.SET will be reset.

### Values

True = Send fail safe configurations to device  
False = No action

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:FS.SET	Boolean	N/A	Write Only

**8 Digital Output Module**

Digital outputs are used to turn loads on and off. Loads may be lights, motors, solenoids or any type of On/Off device.

Digital outputs in the OptiLogic series come in three types: relay, transistor and solid state relay. Each type has applications it is best suited for. For more information, refer to the module manual.

If there is a loss of communication with the host, the output module will enter a fail safe state. There are three types of states: fail safe to all outputs off, fail safe to a pattern and fail safe to last state. The time in which a fail safe state is entered after loss of communication can be delayed up to 25.5 seconds. If certain points need to be treated differently in fail safe mode, a pattern can be used to signify which points are turned on and which ones are turned off in fail safe mode.

**Subtypes**

OL2108, OL2109, OL2111

**Address Types**

[8 Digital Output](#)

[Fail Safe Type](#)

[Fail Safe Pattern](#)

[Fail Safe Time](#)

[Fail Safe Set](#)

**8 Digital Output Addressing Specifications**

Syntax	Data Type	Range	Access
S<slot>:DO<point>	Boolean	0-7	Read/Write
S<slot>:DO<offset>*	Byte	0	Read/Write
S<slot>:DO<offset>*	Word, Short	0	Read/Write
S<slot>:DO<offset>*	DWord, Long	0	Read/Write

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

**Examples**

Address	Value	Description
S1:DO0	1	Slot 1, point 0 (turn point 0 on).
S1:DO0 (as Byte)	255	Slot 1, byte 0 (turn points 0-7 on).
S1:DO0 (as Word)	0	Slot 1, word 0 (turn points 0-7 off).
S1:DO0 (as DWord)	255	Slot 1, DWord 0 (turn points 0-7 on).

**Fail Safe Type Addressing**

There are three types of fail safe states: fail safe to all outputs off, fail safe to a pattern and fail safe to last state. These types may be specified by referencing FS.TYPE.

**Values**

- 1 = Fail safe to all outputs off
- 2 = Fail safe to the pattern contained in FS<point>.PATTERN
- 3 = Fail safe to last state

**Requirements**

None

**Important:** The value specified in FS.TYPE is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.TYPE	Byte, Word, Short, DWord, Long	N/A	Write Only

### Fail Safe Pattern Addressing

Using a fail safe pattern, users can specify which points are set and which ones are cleared when in fail safe mode.

### Values

True = turn point on  
False = turn point off

### Requirements

FS.TYPE must be set to 2 for device to fail safe to a pattern.

**Note:** If using Byte, Word, Short, DWord or Long data types, the bit pattern determines which points are turned on and which ones are turned off.

**Important:** The pattern specified in FS.PATTERN is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>FS<point>.PATTERN	Boolean	0-7	Write Only
S<slot>:FS<offset>.PATTERN*	Byte	0	Write Only
S<slot>:FS<offset>.PATTERN*	Word, Short	0	Write Only
S<slot>:FS<offset>.PATTERN*	DWord, Long	0	Write Only

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

### Examples

Address	Value	Description
S1:FS0.PATTERN	1	Slot 1, turn point 0 on in fail safe mode.*
S1:FS0.PATTERN (as Byte)	255	Slot 1, turn points 0-7 on in fail safe mode.*
S1:FS0.PATTERN (as Word)	0	Slot 1, turn point 0-7 off in fail safe mode.*
S1:FS0.PATTERN (as DWORD)	255	Slot 1, turn points 0-7 on in fail safe mode.*

\*See Notes and Requirements above.

### Fail Safe Time Addressing

After communication between the host and the RTU has been lost, a time delay may be introduced before fail safe mode is engaged for the output module. This time delay can be entered by referencing FS.TIME and is specified in tenths of seconds.

### Requirements

None

**Important:** The value specified in FS.TYPE is not sent to the device until FS.SET is set.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.TIME	Byte, Word, Short, DWord, Long	N/A	Write Only

### Examples

Address	Value	Description
S1:FS.TIME	255	Slot 1, fail safe time delay = 25.5 seconds.
S1:FS.TIME	0	Slot 1, no delay.
S1:FS.TIME	1	Slot 1, fail safe time delay = .1 seconds.

### Fail Safe Set Addressing

For any of the fail safe configuration parameters (type, time and pattern) to be sent to the device, FS.SET must be set. Immediately after the parameters are sent, FS.SET will be reset.

### Values

True = Send fail safe configurations to device  
False = No action

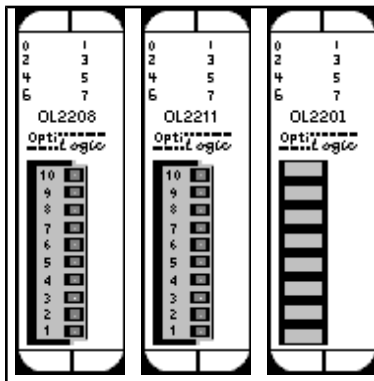
### Requirements

None

### Specifications

Syntax	Data Type	Range	Access
S<slot>:FS.SET	Boolean	N/A	Write Only

## 8 Digital Input Module



Digital input modules are used to monitor the state of discrete field devices. Typical digital inputs are connected to switches, buttons, digital outputs from other equipment, discrete level sensors, thermostats and other On/Off sensing devices.

Digital status is sensed by a controller such as an Opti-Logic system, by passing current through an input sensor. When the current is on, the input state is active. When there is no current, the state is inactive.

### Subtypes

OL2208, OL2211, OL2201

### 8 Digital Input Addressing Specifications

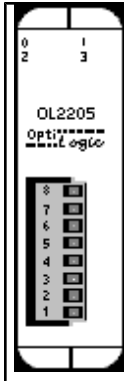
Syntax	Data Type	Range	Access
S<slot>:DI<point>	Boolean	0-7	Read Only
S<slot>:DI<offset>*	Byte	0	Read Only
S<slot>:DI<offset>*	Word, Short	0	Read Only
S<slot>:DI<offset>*	DWord, Long	0	Read Only

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

### Examples

Address	Value	Description
S1:DI0	1	Slot 1, point 0 is on.
S1:DI0 (as Byte)	255	Slot 1, byte 0 (points 0-7 are on).
S1:DI0 (as Word)	0	Slot 1, word 0 (points 0-7 are off).
S1:DI0 (as DWORD)	255	Slot 1, DWord 0 (points 0-7 are on).

## 4 Digital Input Module



Digital input modules are used to monitor the state of discrete field devices. Typical digital inputs are connected to switches, buttons, digital outputs from other equipment, discrete level sensors, thermostats and other On/Off sensing devices.

Digital status is sensed by a controller such as an OptiLogic system, by passing current through an input sensor. When the current is on, the input state is active. When there is no current, the state is inactive.

### Subtypes

OL2205

### 4 Digital Input Addressing Specifications

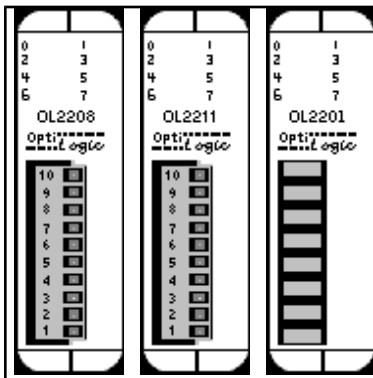
Syntax	Data Type	Range	Access
S<slot>:DI<point>	Boolean	0-3	Read Only
S<slot>:DI<offset>*	Byte	0	Read Only
S<slot>:DI<offset>*	Word, Short	0	Read Only
S<slot>:DI<offset>*	DWord, Long	0	Read Only

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

### Examples

Address	Value	Description
S1:DI0	1	Slot 1, point 0 is on.
S1:DI0 (as Byte)	15	Slot 1, byte 0 (points 0-3 are on).
S1:DI0 (as Word)	0	Slot 1, word 0 (points 0-3 are off).
S1:DI0 (as DWORD)	15	Slot 1, DWord 0 (points 0-3 are on).

## 8 Digital Input Module



Digital input modules are used to monitor the state of discrete field devices. Typical digital inputs are connected to switches, buttons, digital outputs from other equipment, discrete level sensors, thermostats and other On/Off sensing devices.

Digital status is sensed by a controller such as an OptiLogic system, by passing current through an input sensor. When the current is on, the input state is active. When there is no current, the state is inactive.

### Subtypes

OL2208, OL2211, OL2201

### 8 Digital Input Addressing Specifications

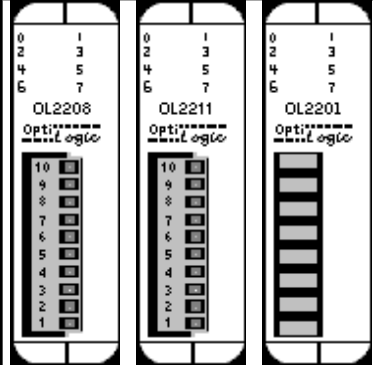
Syntax	Data Type	Range	Access
S<slot>:DI<point>	Boolean	0-7	Read Only
S<slot>:DI<offset>*	Byte	0	Read Only
S<slot>:DI<offset>*	Word, Short	0	Read Only
S<slot>:DI<offset>*	DWord, Long	0	Read Only

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

**Examples**

Address	Value	Description
S1:DI0	1	Slot 1, point 0 is on.
S1:DI0 (as Byte)	255	Slot 1, byte 0 (points 0-7 are on).
S1:DI0 (as Word)	0	Slot 1, word 0 (points 0-7 are off).
S1:DI0 (as DWORD)	255	Slot 1, DWord 0 (points 0-7 are on).

**8 Digital Input Module**



Digital input modules are used to monitor the state of discrete field devices. Typical digital inputs are connected to switches, buttons, digital outputs from other equipment, discrete level sensors, thermostats and other On/Off sensing devices.

Digital status is sensed by a controller such as an Opti-Logic system, by passing current through an input sensor. When the current is on, the input state is active. When there is no current, the state is inactive.

**Subtypes**

OL2208, OL2211, OL2201

**8 Digital Input Addressing Specifications**

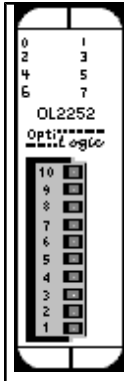
Syntax	Data Type	Range	Access
S<slot>:DI<point>	Boolean	0-7	Read Only
S<slot>:DI<offset>*	Byte	0	Read Only
S<slot>:DI<offset>*	Word, Short	0	Read Only
S<slot>:DI<offset>*	DWord, Long	0	Read Only

\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

**Examples**

Address	Value	Description
S1:DI0	1	Slot 1, point 0 is on.
S1:DI0 (as Byte)	255	Slot 1, byte 0 (points 0-7 are on).
S1:DI0 (as Word)	0	Slot 1, word 0 (points 0-7 are off).
S1:DI0 (as DWORD)	255	Slot 1, DWord 0 (points 0-7 are on).

**2 Channel High Speed Counter Module**



The OL2252 Dual High Speed Pulse Counter module has two 0-15 kHz pulse counter inputs. Each input is independent of the other. There are a number of configuration options available. See OL2252 manual for details. Of the 10 inputs, 6 can be configured for general-purpose input and is referred to in this driver as CDI (counter digital input). Both channels have a software reset and an optional hardware reset as well as a software enable and optional hardware enable.

### Subtypes

OL2252

### Address Types

[Channel Pulse Count](#)

[Channel Enable \(Software\)](#)

[Channel Reset \(Software\)](#)

[Channel Enable \(Hardware\)](#)

[Channel Reset \(Hardware\)](#)

[Channel Debounce Count](#)

[8 Digital Input](#)

### Channel Pulse Count Addressing

The pulse count for each channel can be referenced using address type C<channel>.COUNT.

### Requirements

Channel counter must be enabled either by software or hardware for counter to pulse count.

### Specifications

Syntax	Data Type	Range	Access
S<slot>:C<channel>.COUNT	DWord, Long	1-2	Read Only

### Examples

Address	Value	Description
S1:C1.COUNT	0	Slot 1, channel 1 pulse count is 0.*
S1:C2.COUNT	1000	Slot 1, channel 2 pulse count is 1000.*

\*See Requirements above.

### Channel Enable (Software) Addressing

Channels can be enabled/disabled through software by referencing address type C<channel>.EN.

### Values

True = channel enabled  
False = channel disabled\*

\*Provided channel is not currently hardware enabled.

### Requirements

None

### Specifications

Address	Value	Description
S1:C1.COUNT	0	Slot 1, channel 1 pulse count is 0.*
S1:C2.COUNT	1000	Slot 1, channel 2 pulse count is 1000.*

### Examples

Address	Value	Description
S1:C1.EN	0	Slot 1, ch1 not software enabled.
S1:C2.EN	1	Slot 1, ch2 software enabled.

### Channel Reset (Software) Addressing

A channel's pulse count can be reset through software by referencing address type C<channel>.RES.

#### Values

True = channel pulse count reset  
False = channel pulse counting\*

\*Provided channel is not currently in a hardware reset.

#### Requirements

None

#### Specifications

Syntax	Data Type	Range	Access
S<slot>:C<channel>.RES	Boolean	1-2	Write Only

#### Examples

Address	Value	Description
S1:C1.RES	0	Slot 1, ch1 not software reset.
S1:C2.RES	1	Slot 1, ch2 software reset.

### Channel Enable (Hardware) Addressing

Channels can be enabled by an external enable signal. Enable/disable this capability by referencing address type CCFG<channel>.HEN.

#### Values

True = allow hardware enable of channel  
False = don't allow hardware enable of channel

#### Requirements

None

#### Specifications

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.HEN	Boolean	1-2	Write Only

#### Examples

Address	Value	Description
S1:CCFG1.HEN	0	Slot 1, ch1 hardware enable not allowed.
S1:CCFG2.HEN	1	Slot 1, ch2 hardware enabled allowed.

### Channel Reset (Hardware) Addressing

A channel's pulse count can be reset by an external reset signal. Enable/disable this capability by referencing address type CCFG<channel>.HRES.

#### Values

True = allow hardware reset of pulse count  
False = don't allow hardware reset of pulse count

#### Requirements

None

#### Specifications

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.HRES	Boolean	1-2	Write Only

#### Examples



Address	Value	Description
S1:CCFG1.HRES	0	Slot 1, ch1 hardware reset not allowed.
S1:CCFG2.HRES	1	Slot 1, ch2 hardware reset allowed.

### Channel Debounce Count Addressing

CCFG<channel>.DBNC establishes the maximum pulse frequency that a channel will count.

#### Values

2 = 15 kHz  
 4 = 10 kHz  
 8 = 5 kHz  
 16 = 2.5 kHz  
 40 = 1 kHz

**Note:** The default value is 15 kHz.

#### Requirements

None

#### Specifications

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.DBNC	Byte, Word, Short, DWord, Long	1-2	Write Only

#### Examples

Address	Value	Description
S1:CCFG1.DBNC	40	Slot 1, ch1 max pulse freq set to 1 kHz.
S1:CCFG2.DBNC	2	Slot 1, ch2 max pulse freq set to 15 kHz.

### 8 Digital Input Addressing

The counter's general-purpose inputs are referenced using address type CDI.

#### Requirements

None

#### Specifications

Syntax	Data Type	Range	Access
S<slot>:CDI<point>	Boolean	0-7	Read Only
S<slot>:CDI<offset>*	Byte, Word, Short, DWord, Long	0	Read Only


\*Multiple points can be referenced through the use of this optional syntax. For more information, refer to [Digital I/O Bit Mapping](#).

#### Examples

Address	Value	Description
S1:CDI0	1	Slot 1, point 0 (terminal 10) is on.
S1:CDI0 (as Byte)	0	Slot 1, byte 0 (points 0-7 are all off).

### High Speed Counter Module

---



The OL2258 High Speed Pulse Counter module provides for direct pulse counting for a variety of high-speed pulse interface applications. Typical applications include motion control, metering and velocity measurement.

The OL2258 can be configured to operate in one of three modes:

1. Pulse & Direction (up to 80 kHz input).
2. Up/Down Count (up to 80 kHz input).
3. Quadrature (up to 160 kHz input).

Besides the counter's current count value, the pulse count in the most recent frequency period (user configurable) is also accessible. The OL2258 also contains 2 digital outputs, triggered on when the pulse count reaches the output's minimum range and triggered off when it reaches the output's maximum range.

**Subtypes**

OL2258

**Address Types**

- [Channel Pulse Count](#)
- [Channel Frequency Data](#)
- [Channel Status: A](#)
- [Channel Status: B](#)
- [Channel Status: Z](#)
- [Channel Status: LS](#)
- [Channel Configuration: Count Type](#)
- [Channel Configuration: Frequency Period](#)
- [Channel Configuration: Preset](#)
- [Channel Configuration: Force Preset](#)
- [Channel Configuration: Hold Count](#)
- [Channel Configuration: Z Preset Enable](#)
- [Channel Configuration: LS Preset Enable](#)
- [Triggered Digital Outputs](#)
- [Triggered Digital Outputs: Minimum Range](#)
- [Triggered Digital Outputs: Maximum Range](#)
- [Triggered Digital Outputs: Range Enable](#)

**Channel Pulse Count**

The pulse count for each channel can be referenced using address type C<channel>.COUNT.

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:C<channel>.COUNT	DWord, Long	1	Read Only

**Examples**

Address	Value	Description
S1:C1.COUNT	0	Slot 1, channel 1 pulse count is 0.
S1:C1.COUNT	1000	Slot 1, channel 1 pulse count is 1000.

**Channel Frequency Data**

A pulse count over a preset period of time (C<channel>.COUNT @ (start + period)-C<channel>.COUNT @ start) can be accessed through C<channel>.FREQDATA.

**Requirements**

C<channel>.FREQPER determines the period and should be set before accessing C<channel>.FREQDATA.

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:C<channel>.FREQDATA	Word, Short, DWord, Long	1	Read Only

### Examples

Address	Value	Description
S1:C1.FREQDATA	60000	Slot 1, ch1, if FREQPER = 1 sec, pulse is 60kHz.
S1:C1.FREQDATA	16000	Slot 1, ch1, if FREQPER = 200ms, pulse is 80kHz.

### Channel Status: A

Pulse input A can be referenced through CSTS<channel>.A. Count type defines the meaning of this input as follows:

Count Type	Input Definition
Pulse & Direction	Pulse Input
Up/Down Count	Up Pulse Input
Quadrature Encoder Input	Pulse Input

**Note:** For more information, refer to OptiLogic I/O Modules Manual.

### Values

True = Pulse level high

False = Pulse level low

### Requirements

CCFG<channel>.COUNTTYPE determines the meaning of this input and should be set before accessing CSTS<channel>.A

### Specifications

Syntax	Data Type	Range	Access
S<slot>:CSTS<channel>.A	Boolean	1	Read Only

### Channel Status: B

Pulse input B can be referenced through CSTS<channel>.B. Count type defines the meaning of this input as follows:

Count Type	Input Definition
Pulse & Direction	Direction Input
Up/Down Count	Down Pulse Input
Quadrature Encoder Input	Pulse Input

**Note:** For more information, refer to OptiLogic I/O Modules Manual.

### Values

True = Pulse level high

False = Pulse level low

### Requirements

CCFG<channel>.COUNTTYPE determines the meaning of this input and should be set before accessing CSTS<channel>.B

### Specifications

Syntax	Data Type	Range	Access
S<slot>:CSTS<channel>.B	Boolean	1	Read Only

### Channel Status: Z

Optional input Z provides a means of automatically resetting the count value to a user-defined preset value. Reference Z through CSTS<channel>.Z.

### Values

True = If ZPRESETEN set, send PRESET to device. Otherwise, take no action.

False = No action

**Requirements**

CCFG<channel>.ZPRESETEN must be set in order to enable use of Z in presetting counter

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CSTS<channel>.Z	Boolean	1	Read Only

**Channel Status: LS**

Optional limit switch input LS provides a means of automatically resetting the count value to a user-defined pre-set value. Reference LS through CSTS<channel>.LS.

**Values**

True = If LSPRESETEN set, send PRESET to device. Otherwise, take no action.

False = No action

**Requirements**

CCFG<channel>.LSPRESETEN must be set in order to enable use of LS in presetting counter

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CSTS<channel>.LS	Boolean	1	Read Only

**Channel Configuration: Count Type (Mode)**

The mode of counter operation is configured via CCFG<channel>.COUNTTYPE.

**Values**

0 = Pulse & Direction

1 = Up/Down Count

2 = Quadrature

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.COUNTTYPE	Byte, Word, Short, DWord, Long	1	Read/Write

**Note:** For more information, refer to OptiLogic I/O Modules Manual.

**Examples**

Address	Value	Description
S1:CCFG1.COUNTTYPE	1	Slot 1, ch1 count mode set for pulse & dir.
S1:CCFG1.COUNTTYPE	3	Slot 1, ch1 count mode set for quadrature.

**Channel Configuration: Frequency Period**

Recall that a pulse count over a preset period of time can be accessed through C<channel>.FREQDATA. The period in the FREQDATA formula C<channel>.COUNT @ (start + period)-C<channel>.COUNT @ start is configured through CCFG<channel>.FREQPER.

**Values**

0 = 1 second count

1 = 200 msec count

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.FREQPER	Byte, Word, Short, DWord, Long	1	Read/Write

**Examples**

Address	Value	Description
S1:CCFG1.FREQPER	0	Slot 1, ch1, FREQDATA reflects count in 1 second time window.
S1:CCFG1.FREQPER	1	Slot 1, ch1, FREQDATA reflects count in 200 ms time window.

### Channel Configuration: Preset

Establish a preset count value by setting CCFG<channel>.PRESET.

#### Requirements

For preset to take effect, at least one of the following must be true:  
 CCFG<channel>.FORCEPRESET be set to TRUE  
 CCFG<channel>.ZPRESETEN be set to TRUE and Z input be TRUE  
 CCFG<channel>.LSPRESETEN be set to TRUE and LS input be TRUE

#### Specifications

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.PRESET	DWord, Long	1	Write Only

#### Examples

Address	Value	Description
S1:CCFG1.PRESET	1000	Slot 1, ch1, set preset to 1000.
S1:CCFG1.PRESET	0	Slot 1, ch1, set preset to 0 (default).

### Channel Configuration: Force Preset

Set the counter's count value to the preset given in CCFG<channel>.PRESET.

#### Values

True = Send PRESET to device  
 False = No action

#### Requirements

None

#### Specifications

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.FORCEPRESET	Boolean	1	Write Only

### Channel Configuration: Hold Count

Hold the current count value by referencing CCFG<channel>.HOLDCOUNT.

#### Values

True = Hold count value  
 False = No action

#### Requirements

None.

#### Specifications

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.HOLDCOUNT	Boolean	1	Write Only

### Channel Configuration: Z Preset Enable

Enables Z input to control the presetting of the counter.

#### Values

True = Z input state configured to preset counter when applicable  
 False = Z input state has no control over presetting counter

#### Requirements

None.

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.ZPRESETEN	Boolean	1	Write Only

**Channel Configuration: LS Preset Enable**

Enables LS input to control the presetting of the counter.

**Values**

True = LS input state configured to preset counter when applicable

False = LS input state has no control over presetting counter

**Requirements**

None.

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CCFG<channel>.LSPRESETEN	Boolean	1	Write Only

**Triggered Digital Outputs**

There are 2 digital outputs that can be configured to turn on when the counter count is within a specific range.

The status of whether the digital output is on or off can be accessed through CTRIGDO<output>.

**Requirements**

CTRIGDO outputs must have a range specified and enabled, otherwise outputs will remain off.

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CTRIGDO<output>	Boolean	1-2	Read Only

**Examples**

Address	Value	Description
S1:CTRIGDO1	1	Slot 1, output 1 is on.
S1:CTRIGDO2	0	Slot 1, output 2 is off.

**Triggered Digital Outputs: Minimum Range**

Set the range's minimum value to be the counter's count value at which users desire the CTRIGDO<output> to turn from off to on. This minimum range value is referenced as CTRIGDO<output>.MINRANGE.

**Requirements**

CTRIGDO range usage must be enabled.

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CTRIGDO<output>.MINRANGE	DWord, Long	1-2	Write Only

**Examples**

Address	Value	Description
S1:CTRIGDO1.MINRANGE	100	Slot 1, output 1, turn on when count = 100.
S1:CTRIGDO2.MINRANGE	65536	Slot 1, output 2, turn on when count = 65536.

**Triggered Digital Outputs: Maximum Range**

Set the range's maximum value to be the counter's count value at which users desire the CTRIGDO<output> to turn from on to off. This maximum range value is referenced as CTRIGDO<output>.MAXRANGE.

**Requirements**

CTRIGDO usage range must be enabled.

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CTRIGDO<output>.MAXRANGE	DWord, Long	1-2	Write Only

**Examples**

Address	Value	Description
S1:CTRIGDO1.MAXRANGE	1000	Slot 1, output 1, turn off when count = 1000.
S1:CTRIGDO2.MAXRANGE	0	Slot 1, output 2, turn off when count = 0 (from rollover).

**Triggered Digital Outputs: Range Enable**

Enable the minimum and maximum range values for CTRIGDO (enable usage of a CTRIGDO output) by referencing CTRIGDO<output>.RANGEEN.

**Requirements**

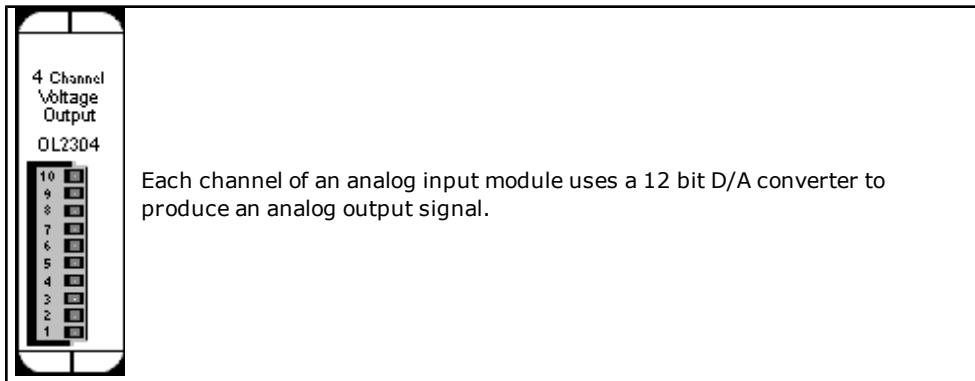
None.

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:CTRIGDO<output>.RANGEEN	Boolean	1-2	Write Only

**Examples**

Address	Value	Description
S1:CTRIGDO1.RANGEEN	1	Slot 1, output 1 enabled.
S1:CTRIGDO2.RANGEEN	0	Slot 1, output 2 disabled.

**4 Channel Analog Output Module****Subtypes**

OL2304

**Address Types**

[4 Channel Analog Output](#)

[4 Channel Analog Output Range](#)

**4 Channel Analog Output Addressing Requirements**

As a precaution, the analog output range should be set prior to writing to an analog output channel.

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:AO<channel>	Word, Short, DWord, Long	1-4	Write Only

**Examples**

Address	Value	Description
S1:A02	1024 *	Slot 1, channel 2.
S1:A04	0 *	Slot 1, channel 4.

\*The exact analog output value depends on voltage range, accuracy and so forth.

**4 Channel Analog Output Range Addressing**

Each channel must be configured for a specific analog output range. This is done by setting AO<channel>.RANGE.

**Values**

- 0 = 0-5 V
- 1 = 0-10 V
- 2 = +/- 5 V
- 3 = +/- 10 V

**Requirements**

None.

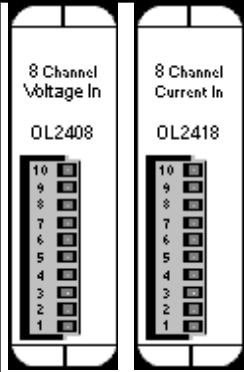
**Specifications**

Syntax	Data Type	Range	Access
S<slot>:AO<channel>.RANGE	Word, Short, DWord, Long	1-4	Write Only

**Examples**

Address	Value	Description
S1:AO2.RANGE	2	Slot 1, channel 2, set voltage range to +/- 5V.
S1:AO4.RANGE	0	Slot 1, channel 4, set voltage range to 0-5V.

**8 Channel Analog Input Module**



Analog inputs are used to monitor the value of continuously variable field measurements. Typical analog inputs are measurements of temperature, pressure, weight, liquid level, pH, flow rate and many other "real world" parameters.

Each channel of an analog input module takes an analog signal as input and uses a 12-bit A/D converter to put that analog value in digital form.

**Subtypes**

OL2408, OL2418

**8 Channel Analog Input Addressing Specifications**

Syntax	Data Type	Range	Access
S<slot>:AI<channel>	Word, Short, DWord, Long	1-8	Read Only

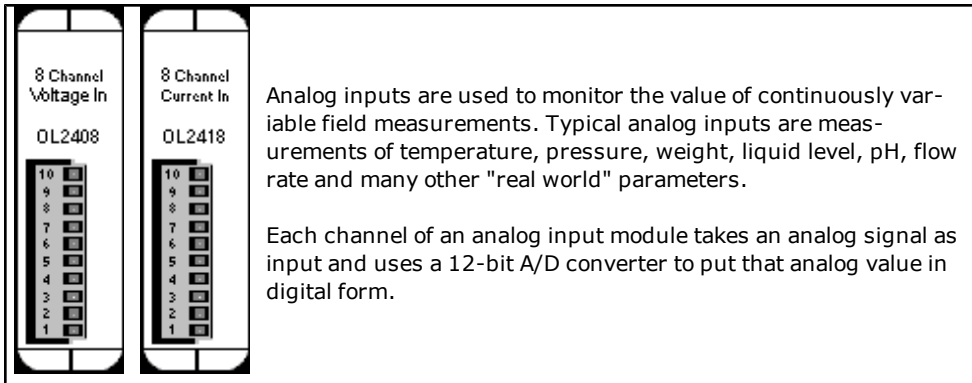
**Examples**

Address	Value	Description
S1:AI2	*	Slot 1, channel 2.
S1:AI8	*	Slot 1, channel 8.

\*Values depend on input level, voltage/current ranges, accuracy and so forth.

**8 Channel Analog Input Module**





### Subtypes

OL2408, OL2418

### 8 Channel Analog Input Addressing Specifications

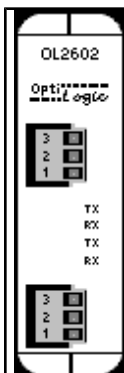
Syntax	Data Type	Range	Access
S<slot>:AI<channel>	Word, Short, DWord, Long	1-8	Read Only

### Examples

Address	Value	Description
S1:AI2	*	Slot 1, channel 2.
S1:AI8	*	Slot 1, channel 8.

\*Values depend on input level, voltage/current ranges, accuracy and so forth.

### Dual RS232 Port Module



The OL2602 is a dual RS232 port serial module. Both the transmit buffer and receive buffer of the driver are 48 bytes in size. Likewise, the corresponding tags can be a maximum of 48 bytes. Incoming bytes are appended to the receive buffer as long as they are received in proper time. This time period depends on the baud rate and is based on a 20-character delay using a 20 ms resolution.

1200 Baud => 160 ms

2400 Baud => 80 ms

4800 Baud => 40 ms

9600 Baud => 20 ms

19200 Baud => 20 ms

For a 4800 baud link, bytes would be appended to the receive buffer as long as they

were received within 40 ms of the last byte sent. After 40 ms, any incoming bytes are treated as a new stream. The receive buffer would clear and only contain these new bytes.

If the receive buffer is full and additional bytes are received within the proper time frame, the buffer will reset with these additional bytes. The first 48 bytes will be lost.

Below is a list of possible configurations:

1. Baud rates: 1200, 2400, 4800, 9600 and 19200.
2. Data bits: 7 or 8
3. Parity: none, odd or even
4. Stop bits: 1, 1.5 or 2

If using the OL4058 RTU, a maximum of one OL2602 modules may be used and it must be placed in slot 0. For the OL4054 RTU, two OL2602 may be used and they must be in either slot 0 or 1.

Port 1 is the top terminal strip, Port 2 is the bottom terminal strip. Below is the pin assignments:

- 1: Signal ground
- 2: TX
- 3: RX

**Subtypes**

OL2602

**Address Types**

- [Serial Input: Data](#)
- [Serial Input: Number of Received Bytes](#)
- [Serial Input: Parity Error](#)
- [Serial Output: Data](#)
- [Serial Output: Number of Bytes Sent](#)
- [Serial Port Configuration: Baud Rate](#)
- [Serial Port Configuration: #Data Bits](#)
- [Serial Port Configuration: Parity](#)
- [Serial Port Configuration: #Stop Bits](#)
- [Serial Port Configuration: Set](#)

**Serial Input: Data Addressing**

To receive serial data, reference address type SI<port>.DATA.

**Note:** The default configuration parameters are 9600, n, 8 and 1.

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:SI<port>.DATA [r][c]*	Byte Array, Char Array	1-2	Read Only
S<slot>:SI<port>.DATA	String	1-2	Read Only

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

**Examples**

Address	Value	Description
S0:SI1.DATA	"hello"	Slot 0, port 1 input data viewed as a string.
S0:SI2.DATA [2][2]	[105, 105] [105, 105]	Slot 0, port 2 input data in array form. In string form this would equate to "iiii".

**Serial Input: Number of Received Bytes Addressing**

The number of received serial bytes (number of bytes in SI<port>.DATA) can be accessed by referencing address type SI<port>.NUMBYTES. If bytes are received within the timeout period mentioned above and are therefore appended to the input DATA buffer, NUMBYTES will reflect the total number of bytes in the input DATA buffer and not the number of bytes received on an individual block read. NUMBYTES will reset upon receiving a new stream.

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:SI<port>.NUMBYTES	Byte, Word, Short, DWord, Long	1-2	Read Only

**Examples**

Address	Value	Description
S1:SI1.NUMBYTES	0	Slot 1, port 1 input, no bytes in input DATA buffer
S1:SI2.NUMBYTES	5	Slot1, port2 input, 5 bytes in input DATA buffer

**Serial Input: Parity Error Addressing**

Reference SI<port>.PARITYERR to determine whether a parity error occurred on the last block read of the serial input port.

**Values**

True = Parity error occurred

False = No parity error occurred

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:SI<port>.PARITYERR	Boolean	1-2	Read Only

**Examples**

Address	Value	Description
S1:SI1.PARITYERR	0	Slot 1, port 1 input, no error.
S1:SI2.PARITYERR	1	Slot 1, port 2 input, parity error occurred.

**Serial Output: Data Addressing**

To transmit serial data, reference address type SO<port>.DATA.

**Note:** The default configuration parameters are 9600, n, 8 and 1.

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:SO<port>.BYTESENT	Byte, Word, Short, DWord, Long	1-2	Read Only

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

**Examples**

Address	Value	Description
S0:SO1.DATA	"hello"	Slot 0, port 1 output, transmit "hello".
S0:SO2.DATA [2][2]	[105, 105][105, 105]	Slot 0, port 2 output data in array form. In string form this would equate to transmitting "iiii".

**Serial Output: Number of Bytes Sent**

Reference SO<port>.BYTESENT to determine how many bytes were sent on the last transmission. This value is available after transmission of serial data.

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:SO<port>.BYTESENT	Byte, Word, Short, DWord, Long	1-2	Read Only

**Examples**

Address	Value	Description
S1:SO1.BYTESENT	0	Slot 1, port 1 output, no bytes sent on last transmission.
S1:SO1.BYTESENT	47	Slot 1, port 1 output, 47 bytes sent.

**Serial Port Configuration: Baud Rate**

To configure the baud rate for a serial port, reference SCFG&lt;port&gt;.BAUD

**Values**

2 = 1200  
 3 = 2400  
 4 = 4800  
 5 = 9600  
 6 = 19200

**Note:** The default value is 9600 baud.**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:SCFG<port>.BAUD	Byte, Word, Short, DWord, Long	1-2	Write Only

**Examples**

Address	Value	Description
S1:SCFG1.BAUD	4	Slot 1, port 1, baud = 4800.
S1:SCFG2.BAUD	6	Slot 1, port 2, baud = 19200.

**Serial Port Configuration: #Data Bits**

To configure the number of data bits for a serial port, reference SCFG&lt;port&gt;.DATABITS

**Values**

7 or 8

**Note:** The default value is 8 data bits.**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
S<slot>:SCFG<port>.DATABITS	Byte, Word, Short, DWord, Long	1-2	Write Only

**Examples**

Address	Value	Description
S1:SCFG1.DATABITS	7	Slot 1, port 1 configured for 7 data bits.
S1:SCFG2.DATABITS	8	Slot 1, port 2 configured for 8 data bits.

**Serial Port Configuration: Parity**

To configure the parity for a serial port, reference SCFG&lt;port&gt;.BAUD

**Values**

0 = none  
 1 = odd  
 2 = even

**Note:** The default value is none.

### Requirements

None

### Specifications

Syntax	Data Type	Range	Access
S<slot>:SCFG<port>.PARITY	Byte, Word, Short, DWord, Long	1-2	Write Only

### Examples

Address	Value	Description
S1:SCFG1.PARITY	0	Slot 1, port 1 configured for no parity.
S1:SCFG2.PARITY	2	Slot 1, port 2 configured for even parity.

### Serial Port Configuration: #Stop Bits

To configure the number of stop bits for a serial port, reference SCFG<port>.STOPBITS

### Values

1 = 1  
 2 = 2  
 3 = 1.5

**Note:** The default value is 1.

### Requirements

None

### Specifications

Syntax	Data Type	Range	Access
S<slot>:SCFG<port>.STOPBITS	Byte, Word, Short, DWord, Long	1-2	Write Only

### Examples

Address	Value	Description
S1:SCFG1.STOPBITS	1	Slot 1, port 1 configured for 1 stop bit.
S1:SCFG2.STOPBITS	3	Slot 1, port 2 configured for 1.5 stop bits.

### Serial Port Configuration: Set

For any of the serial port configuration parameters (baud, parity and so forth) to be sent to the device, SCFG.SET must be set. Immediately after the parameters are sent, SCFG.SET will be reset.

### Values

True = Send serial port configurations to device  
 False = No action

### Requirements

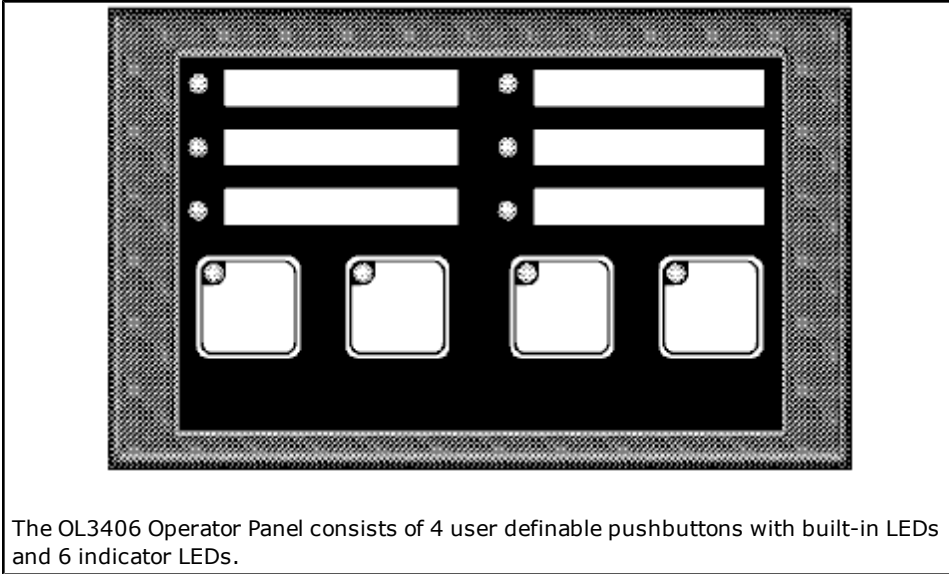
None

### Specifications

Syntax	Data Type	Range	Access
S<slot>:SCFG<port>.SET	Boolean	N/A	Write Only

## OL3406 Operator Panel

---



The OL3406 Operator Panel consists of 4 user definable pushbuttons with built-in LEDs and 6 indicator LEDs.

### Subtypes

OL3406

### Address Types

[Pushbutton LED On State](#)  
[Pushbutton LED Flash State](#)  
[Pushbutton LED Separation State](#)  
[Pushbutton State](#)  
[Pushbutton Configuration](#)  
[Force Pushbutton State](#)  
[Indicator LED On State](#)  
[Indicator LED Flash State](#)

### Pushbutton LED On State Addressing

Each button LED can be forced On/Off without the button being pressed. This can be achieved by referencing address type `BTNLED<button>.ON`.

### Values

True = on  
False = off

### Requirements

Button LED Separation state must be set. Button(s) must also be configured for momentary action. For more information, refer to [PushButton Configuration](#).

### Specifications

Syntax	Data Type	Range	Access
P:BTNLED<button number>.ON	Boolean	0-3	Write Only

### Examples

Address	Value	Description
P:BTNLED0.ON	1	Turn button 0 LED on (left most button).*
P:BTNLED3.ON	1	Turn button 3 LED on (right most button).*

\*See Requirements above.

### Pushbutton LED Flash State Addressing

Each button LED can be forced to flash On/Off without the button being pressed. This can be achieved by referencing address type `BTNLED<button>.FLASH`.

### Values

True = flash on

False = flash off

### Requirements

Button LED ON state must be set for the corresponding button and button LED Separation state must also be set. Button(s) must be configured for momentary action. For more information, refer to [PushButton Configuration Addressing](#).

### Specifications

Syntax	Data Type	Range	Access
P:BTNLED<button number>.FLASH	Boolean	0-3	Write Only

### Examples

Address	Value	Description
P:BTNLED0.FLASH	1	Flash button 0 LED (left most button).*
P:BTNLED3.FLASH	1	Flash button 3 LED (right most button).*

\*See Requirements above.

### Pushbutton LED Separation State Addressing

When LED Separation is set, one is capable of controlling the on and flash state of individual button LEDs (see Pushbutton LED On and Flash State Addressing above).

### Values

True = separation on  
False = separation off

### Requirements

None.

### Specifications

Syntax	Data Type	Range	Access
P:LEDSEP	Boolean	N/A	Read/Write

### Pushbutton State Addressing

Button state (pressed/not pressed) can be monitored by referencing address type BTNSTATUS<button>.

### Values

True = pressed  
False = not pressed

### Requirements

None.

**Note:** Button state depends on the button configuration. For more information, refer to [Pushbutton Configuration Addressing](#).

### Specifications

Syntax	Data Type	Range	Access
P:BTNSTATUS<button number>	Boolean	0-3	Read Only

### Examples

Address	Value	Description
P:BTNSTATUS0	1	Button 0 is pressed (left most button).*
P:BTNSTATUS3	0	Button 3 is not pressed (right most button).

\*See Note above.

### Pushbutton Configuration Addressing

Each button can be configured to either latch their state (alternate action) or hold it momentarily while its being pressed/not pressed.

### Values

True = alternate action  
False = momentary action

### Requirements

None.

### Specifications

Syntax	Data Type	Range	Access
P:BTNCFG<button number>	Boolean	0-3	Read/Write

### Examples

Address	Value	Description
P:BTNCFG0	1	Button 0 is configured for alternate action.
P:BTNCFG3	0	Button 3 is configured for momentary action.

\*See Note above.

### Force Pushbutton State Addressing

Each button can be forced to a desired state. There are three means of forcing button state.

EQUALS:	Desired button state = specified state
OR:	Desired button state = (current state BITWISE OR specified state)
AND:	Desired button state = NOT(current state BITWISE AND specified state)

### Values (Desired button states)

True = button on ("pressed")  
False = button off ("not pressed")

### Requirements

Button must be configured for alternate action.

### Specifications

Below are the three means of forcing button state in detail:

Syntax	Data Type	Range	Access
P:BTNFORCE<button number>.EQUALS	Boolean	0-3	Write Only
P:BTNFORCE<button number>.OR	Boolean	0-3	Write Only
P:BTNFORCE<button number>.AND	Boolean	0-3	Write Only

### Examples

Address	Value	Description
P:BTNFORCE0.EQUALS	1	Force button 0 state to be on.
P:BTNFORCE0.OR	1	OR current state with 1. Force button 0 state to be on.
P:BTNFORCE0.AND	1	If button 0 state is currently on, set state to off. Otherwise, do nothing.

\*See Requirements above.

### Indicator LED On State Addressing

Each indicator LED can be forced On/Off by referencing address type INDLED<led>.ON.

### Values

True = on  
False = off

### Requirements

None.

### Specifications

Syntax	Data Type	Range	Access
P:INDLED<led number>.ON	Boolean	0-5	Write Only



**Examples**

Address	Value	Description
P:INDLED0.ON	1	Turn indicator LED 0 on (upper left LED).
P:INDLED5.ON	1	Turn indicator LED 5 on (bottom right LED).

**Indicator LED Flash State Addressing**

Each indicator LED can be forced to flash On/Off by referencing address type INDLED<led>.FLASH.

**Values**

True = flash on  
False = flash off

**Requirements**

Indicator LED ON state must be set for the corresponding LED.

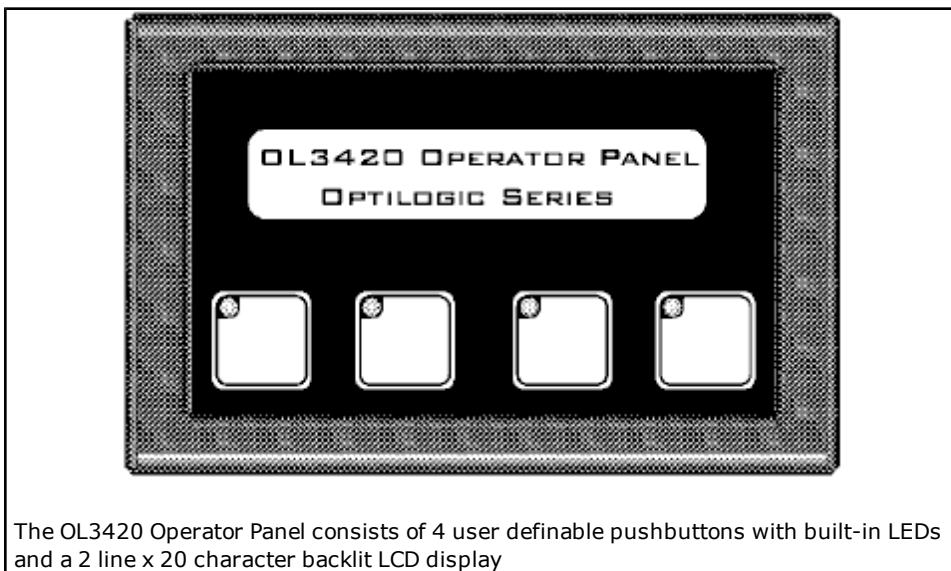
**Specifications**

Syntax	Data Type	Range	Access
P:INDLED<led number>.FLASH	Boolean	0-5	Write Only

**Examples**

Address	Value	Description
P:INDLED1.FLASH	1	Flash indicator LED 1 (upper right LED).*
P:INDLED4.FLASH	1	Flash indicator LED 4 (lower left LED).*

\*See Requirements above.

**OL3420 Operator Panel**

The OL3420 Operator Panel consists of 4 user definable pushbuttons with built-in LEDs and a 2 line x 20 character backlit LCD display

**Subtypes**

OL3420

**Address Types**

[Pushbutton LED On State](#)  
[Pushbutton LED Flash State](#)  
[Pushbutton LED Separation State](#)  
[Pushbutton State](#)  
[Pushbutton Configuration](#)  
[Force Pushbutton State](#)  
[Alphanumeric Display](#)

**Pushbutton LED On State Addressing**

Each button LED can be forced On/Off without the button being pressed. This can be achieved by referencing address type BTNLED<button>.ON.

**Values**

True = on  
False = off

**Requirements**

Button LED Separation state must be set. Button(s) must also be configured for momentary action. For more information, refer to [PushButton Configuration](#).

**Specifications**

Syntax	Data Type	Range	Access
P:BTNLED<button number>.ON	Boolean	0-3	Write Only

**Examples**

Address	Value	Description
P:BTNLED0.ON	1	Turn button 0 LED on (left most button).*
P:BTNLED3.ON	1	Turn button 3 LED on (right most button).*

\*See Requirements above.

**Pushbutton LED Flash State Addressing**

Each button LED can be forced to flash On/Off without the button being pressed. This can be achieved by referencing address type BTNLED<button>.FLASH.

**Values**

True = flash on  
False = flash off

**Requirements**

Button LED ON state must be set for the corresponding button and button LED Separation state must be set. Button(s) must also be configured for momentary action. For more information, refer to [PushButton Configuration](#).

**Specifications**

Syntax	Data Type	Range	Access
P:BTNLED<button number>.FLASH	Boolean	0-3	Write Only

**Examples**

Address	Value	Description
P:BTNLED0.FLASH	1	Flash button 0 LED (left most button).*
P:BTNLED3.FLASH	1	Flash button 3 LED (right most button).*

\*See Requirements above.

**Pushbutton LED Separation State Addressing**

When LED Separation is set, one is capable of controlling the on and flash state of individual button LEDs (see Pushbutton LED On and Flash State Addressing above).

**Values**

True = separation on  
False = separation off

**Requirements**

None.

Syntax	Data Type	Range	Access
P:LEDSEP	Boolean	N/A	Read/Write

**Pushbutton State Addressing**

Button state (pressed/not pressed) can be monitored by referencing address type BTNSTATUS<button>.

### Values

True = pressed  
False = not pressed

### Requirements

None.

**Note:** Button state depends on the button configuration. See [Pushbutton Configuration Addressing](#) below.

### Specifications

Syntax	Data Type	Range	Access
P:BTNSTATUS<button number>	Boolean	0-3	Read Only

### Examples

Address	Value	Description
P:BTNSTATUS0	1	Button 0 is pressed (left most button).*
P:BTNSTATUS3	0	Button 3 is not pressed (right most button).

\*See Note above.

### Pushbutton Configuration Addressing

Each button can be configured to either latch their state (alternate action) or hold it momentarily while its being pressed/not pressed.

### Values

True = alternate action  
False = momentary action

### Requirements

None.

### Specifications

Syntax	Data Type	Range	Access
P:BTNCFG<button number>	Boolean	0-3	Read/Write

### Examples

Address	Value	Description
P:BTNCFG0	1	Button 0 is configured for alternate action.
P:BTNCFG3	0	Button 3 is configured for momentary action.

\*See Note above.

### Force Pushbutton State Addressing

Each button can be forced to a desired state. There are three ways to force button state.

EQUALS:	Desired button state = specified state
OR:	Desired button state = (current state BITWISE OR specified state)
AND:	Desired button state = NOT(current state BITWISE AND specified state)

### Values (Desired button states)

True = button on ("pressed")  
False = button off ("not pressed")

### Requirements

Button must be configured for alternate action.

### Specifications

Below are the three means of forcing button state in detail.

Syntax	Data Type	Range	Access
P:BTNFORCE<button number>.EQUALS	Boolean	0-3	Write Only
P:BTNFORCE<button number>.OR	Boolean	0-3	Write Only
P:BTNFORCE<button number>.AND	Boolean	0-3	Write Only

### Examples

Address	Value	Description
P:BTNFORCE0.EQUALS	1	Force button 0 state to be on.
P:BTNFORCE0.OR	1	OR current state with 1. Force button 0 state to be on.
P:BTNFORCE0.AND	1	If button 0 state is currently on, set state to off. Otherwise, do nothing.

\*See Requirements above.

### Alphanumeric Display Addressing

There are two lines of display for alphanumeric strings of length 20 characters or less.

### Requirements

None.

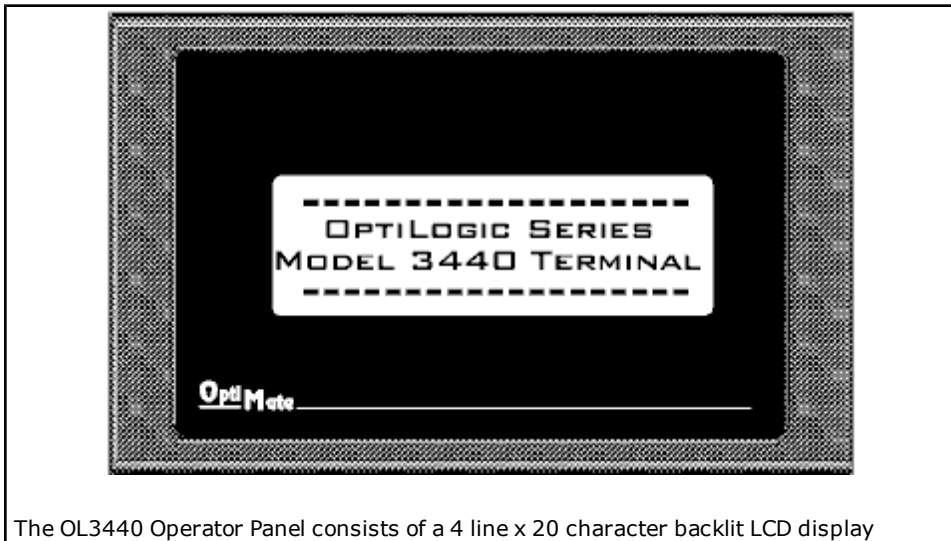
### Specifications

Syntax	Data Type	Range	Access
P:LINE<line number>	String	0-1	Write Only

### Examples

Syntax	Data Type	Range	Access
P:LINE<line number>	String	0-1	Write Only

## OL3440 Operator Panel



The OL3440 Operator Panel consists of a 4 line x 20 character backlit LCD display

### Subtypes

OL3440

### Alphanumeric Display Addressing

There are four lines of display for alphanumeric strings of length 20 characters or less.

### Requirements

None.

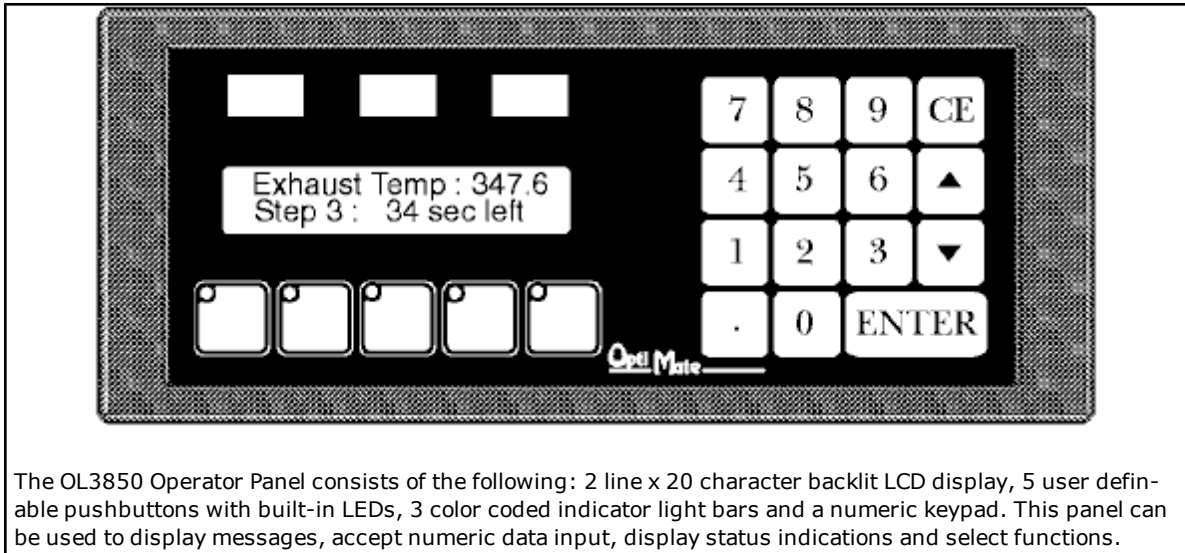
### Specifications

Syntax	Data Type	Range	Access
P:LINE<line number>	String	0-3	Write Only

### Examples

Address	Value	Description
P:LINE0	"hello"	Set line 0 (top line) string to "hello".
P:LINE3	"world"	Set line 3 (bottom line) string to "world".

## OL3850 Operator Panel



The OL3850 Operator Panel consists of the following: 2 line x 20 character backlit LCD display, 5 user definable pushbuttons with built-in LEDs, 3 color coded indicator light bars and a numeric keypad. This panel can be used to display messages, accept numeric data input, display status indications and select functions.

### Subtypes

OL3850

### Address Types

- [Pushbutton LED On State](#)
- [Pushbutton LED Flash State](#)
- [Pushbutton LED Separation State](#)
- [Pushbutton State](#)
- [Pushbutton Configuration](#)
- [Force Pushbutton State](#)
- [Light Bar On State](#)
- [Light Bar Flash State](#)
- [Alphanumeric Display](#)
- [Keypad Data](#)
- [Keypad Data Available](#)
- [Keypad Arrow Max](#)
- [Keypad Arrow Min](#)

### Pushbutton LED On State Addressing

Each button LED can be forced On/Off without the button being pressed. This can be achieved by referencing address type `BTNLED<button>.ON`.

### Values

True = on  
False = off

### Requirements

Button LED Separation state must be set. Button(s) must also be configured for momentary action. For more information, refer to [PushButton Configuration Addressing](#).

### Specifications

Syntax	Data Type	Range	Access
P:BTNLED<button number>.ON	Boolean	0-4	Write Only

### Examples

Address	Value	Description
P:BTNLED0.ON	1	Turn button 0 LED on (left most button).*
P:BTNLED4.ON	1	Turn button 4 LED on (right most button).*

\*See Requirements above.

### Pushbutton LED Flash State Addressing

Each button LED can be forced to flash On/Off without the button being pressed. This can be achieved by referencing address type BTNLED<button>.FLASH.

### Values

True = flash on  
False = flash off

### Requirements

Button LED ON state must be set for the corresponding button and button LED Separation state must be set. Button(s) must also be configured for momentary action. For more information, refer to [PushButton Configuration Addressing](#).

Syntax	Data Type	Range	Access
P:BTNLED<button number>.FLASH	Boolean	0-4	Write Only

### Examples

Address	Value	Description
P:BTNLED0.FLASH	1	Flash button 0 LED (left most button).*
P:BTNLED4.FLASH	1	Flash button 4 LED (right most button).*

\*See Requirements above.

### Pushbutton LED Separation State Addressing

When LED Separation is set, one is capable of controlling the on and flash state of individual button LEDs. For more information, refer to "Pushbutton LED On and Flash State Addressing" above.

### Values

True = separation on  
False = separation off

### Requirements

None.

### Specifications

Syntax	Data Type	Range	Access
P:LEDSEP	Boolean	N/A	Read/Write

### Pushbutton State Addressing

Button state (pressed/not pressed) can be monitored by referencing address type BTNSTATUS<button>.

### Values

True = pressed  
False = not pressed

### Requirements

None.

**Note:** Button state depends on the button configuration. For more information, refer to [Pushbutton Configuration Addressing](#) below.

### Specifications

Syntax	Data Type	Range	Access
P:BTNSTATUS<button number>	Boolean	0-4	Read Only

### Examples

Address	Value	Description
P:BTNSTATUS0	1	Button 0 is pressed (left most button).*
P:BTNSTATUS4	0	Button 4 is not pressed (right most button).

\*See Note above.

### Pushbutton Configuration Addressing

Each button can be configured to either latch their state (alternate action) or hold it momentarily while its being pressed/not pressed.

### Values

True = alternate action  
False = momentary action

### Requirements

None.

### Specifications

Syntax	Data Type	Range	Access
P:BTNCFG<button number>	Boolean	0-4	Read/Write

### Examples

Address	Value	Description
P:BTNCFG0	1	Button 0 is configured for alternate action.
P:BTNCFG4	0	Button 4 is configured for momentary action.

### Force Pushbutton State Addressing

Each button can be forced to a desired state. There are three ways to force button state.

EQUALS:	Desired button state = specified state
OR:	Desired button state = (current state BITWISE OR specified state)
AND:	Desired button state = NOT(current state BITWISE AND specified state)

### Values (Desired button states)

True = button on ("pressed")  
False = button off ("not pressed")

### Requirements

Button must be configured for alternate action.

### Specifications

Below are the three means of forcing button state in detail:

Syntax	Data Type	Range	Access
P:BTNFORCE<button number>.EQUALS	Boolean	0-4	Write Only
P:BTNFORCE<button number>.OR	Boolean	0-4	Write Only
P:BTNFORCE<button number>.AND	Boolean	0-4	Write Only

### Examples

Address	Value	Description
P:BTNFORCE0.EQUALS	1	Force button 0 state to be on.*
P:BTNFORCE0.OR	1	OR current state with 1. Force button 0 state to be on.*
P:BTNFORCE0.AND	1	If button 0 state is currently on, set state to off. otherwise, do nothing.*

\*See Requirements above.

**Light Bar On State Addressing**

Each light bar can be forced On/Off by referencing address type LITEBAR<bar>.ON.

**Values**

True = on  
False = off

**Requirements**

None.

**Specifications**

Syntax	Data Type	Range	Access
P:LITEBAR<light bar>.ON	Boolean	0-2	Write Only

**Examples**

Address	Value	Description
P:LITEBAR0.ON	1	Turn light bar 0 on (left most bar).
P:LITEBAR2.ON	1	Turn light bar 2 on (right most bar).

**Light Bar Flash State Addressing**

Each light bar can be forced to flash On/Off by referencing address type LITEBAR<bar>.FLASH.

**Values**

True = flash on  
False = flash off

**Requirements**

Light Bar ON state must be set for the corresponding light bar.

**Specifications**

Syntax	Data Type	Range	Access
P:LITEBAR<light bar>.FLASH	Boolean	0-2	Write Only

**Examples**

Address	Value	Description
P:LITEBAR0.FLASH	1	Flash light bar 0 LED (left most bar).*
P:LITEBAR2.FLASH	1	Flash light bar 2 LED (right most bar).*

\*See Requirements above.

**Alphanumeric Display Addressing**

There are two lines of display for alphanumeric strings of length 20 characters or less. Keypad data may also be inserted into the text string by using the caret (^) as a placeholder for each digit (including the decimal point) of the keypad data.

**Requirements**

None.

**Specifications**

Syntax	Data Type	Range	Access
P:LINE<line number>	String	0-1	Write Only

**Examples**

Address	Value	Description
P:LINE0	"hello"	Set line 0 (top line) string to "hello".
P:LINE1	"world"	Set line 1 (bottom line) string to "world".
P:LINE0	"^^^^^^"	If keypad data = 1234.56, then line 0 string will be "1234.56".*

\*For more information, refer to "Keypad Data Addressing" below.

**Keypad Data Addressing**



Numeric data can be read from (entered via keypad or this address) and written to (via this address) the panel keypad. Float precision can be specified by appending a bit (0 to 10 allowed) to the KDATA address. This will represent the floating point precision on any writes to the device. If no precision is specified (the default case), the precision will be set such that the number of digits in the integer and fractional parts sums up to 10. If a precision is specified such that the 10-digit limit is exceeded, the precision will be set to the default value previously discussed. Note that the value written to the device may differ from the value displayed in the client. This may be due to floating point round off and truncation errors from the driver, client or both. For more information on floating point precision, refer to [Device Setup](#).

**Note:** Keypad data may be altered using the arrows located on the keypad. Upper and lower bounds set by Arrow Max and Arrow Min respectively, will only limit the data set by the arrows, not the data set in KDATA. Only the data type can place constraints on the upper and lower limits of the keypad data when set using KDATA.

### Specifications

Syntax	Data Type	Range	Access
P:KDATA	<b>Double</b> , Float, DWord	N/A	Read/Write
P:KDATA.<precision>	<b>Double</b> , Float, DWord	N/A	Read/Write

### Examples (Writes)

Address	Value	Description
P.KDATA	10.123456	10.123456 will be written to the device.
P:KDATA.4 as a float	10.123456	10.1235 will be written to the device.
P:KDATA	9999999999	9999999999 will be written to the device.

### Keypad Data Available Addressing

To determine if new keypad data has been entered at the panel, reference KDATAREADY. This flag can be cleared by writing to KDATA.

### Values

True = new data has been entered

False = all data has been read from driver, no new data

Syntax	Data Type	Range	Access
P:KDATAREADY	<b>Boolean</b>	N/A	Read Only

### Keypad Arrow Max Addressing

Panel keypad data entered via the arrows, can be upper bounded by referencing ARROW.MAX. Any keypad data entered above this max will automatically get set to this max value.

**Note:** Initially, before ARROW.MAX is set, the upper limit internally to the device is 9999999999. When the limit is set, keypad data cannot exceed the size of ARROW.MAX which is 32 bits (DWord).

### Specifications

Syntax	Data Type	Range	Access
P:ARROW.MAX	<b>DWord</b>	N/A	Write Only

### Keypad Arrow Min Addressing

Panel keypad data entered via the arrows, can be lower bounded by referencing ARROW.MIN. Any keypad data entered below this min will automatically get set to this min value.

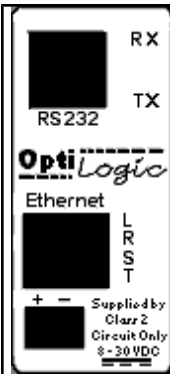
**Note:** Initially, before ARROW.MIN is set, the lower limit internally to the device is 0.

### Specifications

Syntax	Data Type	Range	Access
P:ARROW.MIN	<b>DWord</b>	N/A	Write Only

## Base RS232 Port

---



The Ethernet Base contains an RS232 serial port. Both the transmit buffer and receive buffer of the driver are 48 bytes in size. Likewise, the corresponding tags can be a maximum of 48 bytes. Incoming bytes are appended to the receive buffer as long as they are received in proper time. This time period depends on the baud rate and is based on a 20-character delay using a 20 ms resolution.

300 Baud => 660 ms

1200 Baud => 160 ms

2400 Baud => 80 ms

4800 Baud => 40 ms

9600 Baud => 20 ms

19200 Baud => 20 ms

For a 4800 baud link, bytes would be appended to the receive buffer as long as they were received within 40 ms of the last byte sent. After 40 ms, any incoming bytes are treated as a new stream. The receive buffer would clear and only contain these new bytes.

If the receive buffer is full and additional bytes are received within the proper time frame, the buffer will reset with these additional bytes. The first 48 bytes will be lost.

Below is a list of possible configurations:

1. Baud rates: 300, 1200, 2400, 4800, 9600 and 19200.
2. Data bits: 7 or 8
3. Parity: none, odd or even
4. Stop bits: 1, 1.5 or 2

An RJ45 connector is required.

### Subtypes

OL4054, OL4058

### Address Types

[Serial Input: Data](#)

[Serial Input: Number of Received Bytes](#)

[Serial Input: Parity Error](#)

[Serial Output: Data](#)

[Serial Output: Number of Bytes Sent](#)

[Serial Port Configuration: Baud Rate](#)

[Serial Port Configuration: #Data Bits](#)

[Serial Port Configuration: Parity](#)

[Serial Port Configuration: #Stop Bits](#)

**Serial Port Configuration: Set****Serial Input: Data Addressing**

To receive serial data, reference address type SI<port>.DATA.

**Note:** The default configuration parameters are 300, n, 8 and 1.

**Specifications**

Syntax	Data Type	Range	Access
B:SI<port>.DATA [r][c]*	Byte Array, Char Array	0	Read Only
B:SI<port>.DATA	String	0	Read Only

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

**Examples**

Address	Value	Description
B:SI0.DATA	"hello"	port 0 input data viewed as a string
B:SI0.DATA [2][2]	[105, 105][105, 105]	port 0 input data in array form. In string form this would equate to "iiii"

**Serial Input: Number of Received Bytes Addressing**

The number of received serial bytes (number of bytes in SI<port>.DATA. can be accessed by referencing address type SI<port>.NUMBYTES. If bytes are received within the timeout period mentioned above and are therefore appended to the input DATA buffer, NUMBYTES will reflect the total number of bytes in the input DATA buffer and not the number of bytes received on an individual block read. NUMBYTES will reset upon receiving a new stream.

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
B:SI<port>:NUMBYTES	Byte, Word, Short, DWord, Long	0	Read Only

**Examples**

Address	Value	Description
B:SI0.NUMBYTES	0	port 0 input, no bytes in input DATA buffer
B:SI0.NUMBYTES	5	port 0 input, 5 bytes in input DATA buffer

**Serial Input: Parity Error Addressing**

Reference SI<port>.PARITYERR to determine whether a parity error occurred on the last block read of the serial input port.

**Values**

True = Parity error occurred

False = No parity error occurred

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
B:SI<port>.PARITYERR	Boolean	0	Read Only

**Examples**

Address	Value	Description
B:.SI0.PARITYERR	0	port 0 input, no error
B:SI0.PARITYERR	1	port 0 input, parity error occurred

**Serial Output: Data Addressing**

To transmit serial data, reference address type SO<port>.DATA.

**Note:** The default configuration parameters are 300, n, 8 and 1.

### Specifications

Syntax	Data Type	Range	Access
B:SO<port>.DATA [r][c]*	<b>Byte Array</b> , Char Array	0	Write Only
B:SO<port>.DATA	String	0	Write Only

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

### Examples

Address	Value	Description
B:SO0.DATA	"hello"	port 0 output, transmit "hello"
B:SO0.DATA [2][2]	[105, 105][105, 105]	port 0 output data in array form. In string form this would equate to transmitting "iiii"

### Serial Output: Number of Bytes Sent

Reference SO<port>.BYTESENT to determine how many bytes were sent on the last transmission. This value is available after transmission of serial data.

### Requirements

None

### Specifications

Syntax	Data Type	Range	Access
B:SO<port>.BYTESENT	<b>Byte</b> , Word, Short, DWord, Long	0	Read Only

### Examples

Address	Value	Description
B:SO0.BYTESENT	0	port 0 output, no bytes sent on last transmission
B:SO0.BYTESENT	47	port 0 output, 47 bytes sent

### Serial Port Configuration: Baud Rate

To configure the baud rate for a serial port, reference SCFG<port>.BAUD

### Values

- 1 = 300
- 2 = 1200
- 3 = 2400
- 4 = 4800
- 5 = 9600
- 6 = 19200

**Note:** The default value is 300 baud.

### Requirements

None

### Specifications

Syntax	Data Type	Range	Access
B:SCFG<port>.BAUD	<b>Byte</b> , Word, Short, DWord, Long	0	Write Only

### Examples

Address	Value	Description
B:SCFG0.BAUD	4	port 0, baud = 4800
B:SCFG0.BAUD	6	port 0, baud = 19200

### Serial Port Configuration: #Data Bits

To configure the number of data bits for a serial port, reference SCFG<port>.DATABITS

**Values**

7 or 8

**Note:** The default value is 8 data bits.**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
B:SCFG<port>.DATABITS	Byte, Word, Short, DWord, Long	0	Write Only

**Examples**

Address	Value	Description
B:SCFG0.DATABITS	7	port 0 configured for 7 data bits
B:SCFG0.DATABITS	8	port 0 configured for 8 data bits

**Serial Port Configuration: Parity**

To configure the parity for a serial port, reference SCFG&lt;port&gt;.BAUD

**Values**

0 = none

1 = odd

2 = even

**Note:** The default value is none.**Requirements:**

None

**Specifications**

Syntax	Data Type	Range	Access
B:SCFG<port>.PARITY	Byte, Word, Short, DWord, Long	0	Write Only

**Examples**

Address	Value	Description
B:SCFG0.PARITY	0	port 0 configured for no parity
B:SCFG0.PARITY	2	port 0 configured for even parity

**Serial Port Configuration: #Stop Bits**

To configure the number of stop bits for a serial port, reference SCFG&lt;port&gt;.STOPBITS

**Values**

1 = 1

2 = 2

3 = 1.5

**Note:** The default value is 1.**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
B:SCFG<port>.STOPBITS	Byte, Word, Short, DWord, Long	0	Write Only

**Examples**

Address	Value	Description
B:SCFG0.STOPBITS	1	port 0 configured for 1 stop bit
B:SCFG0.STOPBITS	3	port 0 configured for 1.5 stop bits

**Serial Port Configuration: Set**

For any of the serial port configuration parameters (baud, parity and so forth) to be sent to the device, SCFG.SET must be set. Immediately after the parameters are sent, SCFG.SET will be reset.

**Values**

True = Send serial port configurations to device  
False = No action

**Requirements**

None

**Specifications**

Syntax	Data Type	Range	Access
B:SCFG<port>.SET	Boolean	N/A	Write Only

## Error Descriptions

---

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

### Driver Error Messages

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

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

[Memory allocation error](#)

### Driver Warning Messages

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

[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](#)

[Array size is out of range for address <address>](#)

[Array support is not available for the specified address: <address>](#)

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

[Base w/ type=<type>, Major ver.=<major>, Minor ver.=<minor> is currently not supported. Contact Technical Support](#)

[Module in slot <slot> w/ type=<type>, subtype=<subtype> is currently not supported. Contact Technical Support](#)

[Panel w/ type=<type>, subtype=<subtype> is currently not supported. Contact Technical Support](#)

### Read Errors

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

[Base module referenced in address <address> on device <device name> does not exist](#)

[Slot module referenced in address <address> on device <device name> does not exist](#)

[Panel module referenced in address <address> on device <device name> does not exist](#)

[Address <address> contains an invalid address type for RTU module, device <device name>](#)

[Address <address> contains an invalid address type for module <subtype>, slot <slot>, device <device name>](#)

[Address <address> contains an invalid address type for panel module <subtype>, device <device name>](#)

[Address <address> is out of range for module <subtype> in slot <slot>, device <device name>](#)

[Address <address> is out of range for panel module <subtype>, device <device name>](#)

[Port <port> of the base module returned an error with value = <error value>](#)

[Module <subtype> in slot <slot> returned an error with value = <error value>](#)

[Port <port> of module <subtype> in slot <slot> returned an error with value = <error value>](#)

[Panel module <subtype> returned an error with value = <error value>](#)

### Write Errors

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

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

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

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

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

[Write rejected. Address <address> contains an invalid address type for RTU module, device <device name>](#)

[Write rejected. Address <address> contains an invalid address type for module <subtype>, slot <slot>, device <device name>](#)

[Write rejected. Address <address> contains an invalid address type for panel module <subtype>, device <device name>](#)

[Write rejected. Address <address> is out of range for module <subtype> in slot <slot>, device <device name>](#)

[Write rejected. Address <address> is out of range for panel module <subtype>, device <device name>](#)

[Write failed. Port <port> of the base module returned an error with value = <error value>](#)

[Write failed. Module <subtype> in slot <slot> returned an error with value = <error value>](#)

[Write failed. Port <port> of module <subtype> in slot <slot> returned an error with value = <error value>](#)

[Write failed. Panel module <subtype> returned an error with value = <error value>](#)

## OptiLogic Device Error Codes

Error codes returned from the OptiLogic RTU, as well as their equivalent error string, are as follows.

Error Code	Description
0	No Error
1	Module message was the improper length
2	Improper module command
3	Module not present

## Driver Error Messages

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

### Driver Error Messages

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

[Winsock V1.1 or higher must be installed to use the OptiLogic 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 OptiLogic 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.

### Driver Warning Messages



**Device '<device name>' is not responding****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****Array size is out of range for address <address>****Array support is not available for the specified address: <address>****Data type <type> is not valid for device address <address>****Base w/ type=<type>, Major ver.=<major>, Minor ver.=<minor> is currently not supported. Contact Technical Support****Module in slot <slot> w/ type=<type>, subtype=<subtype> is currently not supported. Contact Technical Support****Panel w/ type=<type>, subtype=<subtype> is currently not supported. Contact Technical Support****Device '<device name>' is 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.
4. The response from the device took longer to receive than the amount of time specified in the "Request Timeout" device setting.

**Solution:**

1. Verify the cabling between the PC and the OptiLogic RTU 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.
4. Increase the Request Timeout setting so that the entire response can be handled.

**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:**

Re-enter 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 in the client application.

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

---

**Error Type:**

Warning

**Possible Cause:**

The target device does not support a tag address that has been specified dynamically.

**Solution:**

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

**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.

**Array size is out of range for address <address>**

---

**Error Type:**

Warning

**Possible Cause:**

A tag address that has been specified dynamically is requesting an array size that is too large for the address type of the driver

**Solution:**

Re-enter the address in the client application to specify a smaller value for the array or a different starting point.

**Array support is not available for the specified address: <address>**

---

**Error Type:**

Warning

**Possible Cause:**

A tag address that has been specified dynamically contains an array reference for an address type that doesn't support arrays.

**Solution:**

Re-enter the address in the client application to remove the array reference or correct the address type.

**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.

**Base with type=<type> Major ver.=<major> Minor ver.=<minor> is currently not supported. Contact Technical Support**

---

**Error Type:**

Warning

**Possible Cause:**

An OptiLogic RTU with type '<type>', version '<major>' and '<minor>' is currently not supported in this driver.

**Solution:**

New RTUs may be released that are not currently handled in the driver. Contact Technical Support so that support can be added for this new RTU.

### **Module in slot <slot> w/ type=<type>, subtype=<subtype> is currently not supported. Contact Technical Support**

---

**Error Type:**

Warning

**Possible Cause:**

An OptiLogic module with type '<type>', version '<major>' and '<minor>' is currently not supported in this driver.

**Solution:**

New modules may be released that are not currently handled in the driver. Contact Technical Support so that support can be added for this new module.

### **Panel with type=<type>, subtype=<subtype> is currently not supported. Contact Technical Support**

---

**Error Type:**

Warning

**Possible Cause:**

An OptiLogic panel with type '<type>', version '<major>' and '<minor>' is currently not supported in this driver.

**Solution:**

New modules may be released that are not currently handled in the driver. Contact Technical Support so that support can be added for this new panel.

### **Read Errors**

---

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

**Read Errors**

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

[Base module referenced in address <address> on device <device name> does not exist](#)

[Slot module referenced in address <address> on device <device name> does not exist](#)

[Panel module referenced in address <address> on device <device name> does not exist](#)

[Address <address> contains an invalid address type for RTU module, device <device name>](#)

[Address <address> contains an invalid address type for module <subtype>, slot <slot>, device <device name>](#)

[Address <address> contains an invalid address type for panel module <subtype>, device <device name>](#)

[Address <address> is out of range for module <subtype> in slot <slot>, device <device name>](#)

[Address <address> is out of range for panel module <subtype>, device <device name>](#)

[Port <port> of the base module returned an error with value = <error value>](#)

[Module <subtype> in slot <slot> returned an error with value = <error value>](#)

[Port <port> of module <subtype> in slot <slot> returned an error with value = <error value>](#)

[Panel module <subtype> returned an error with value = <error value>](#)

### **Frame received from device <device name> contains errors**

---

**Error Type:**

Warning

**Possible Cause:**

The OptiLogic 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.

**Base module referenced in address <address> on device <device name> does not exist**

---

**Error Type:**

Warning

**Possible Cause:**

A tag address references an RTU that is currently not supported.

**Solution:**

1. Verify that the address is correct; if it is not, re-enter it in the client application.
2. Contact Technical Support regarding the RTU that is not supported.

**Slot module referenced in address <address> on device <device name> does not exist**

---

**Error Type:**

Warning

**Possible Cause:**

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

**Solution:**

1. Verify that the address is correct; if it is not, re-enter it in the client application.
2. Verify that the module fits securely into its mating connector on the motherboard.

**Panel module referenced in address <address> on device <device name> does not exist**

---

**Error Type:**

Warning

**Possible Cause:**

A tag address references a panel that is not connected to the RTU.

**Solution:**

1. Verify that the address is correct; if it is not, re-enter it in the client application.
2. Verify that both ends of the interconnect cable are securely connected.

**Address <address> contains an invalid address type for RTU module, device <device name>**

---

**Error Type:**

Warning

**Possible Cause:**

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

**Solution:**

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

**Address <address> contains an invalid address type for module <subtype>, slot <slot>, device <device name>**

---

**Error Type:**

Warning

**Possible Cause:**

A tag references an address using an address type that is not valid for module '<subtype>' in slot '<slot>'.

**Solution:**

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

---

**Address <address> contains an invalid address type for panel module <subtype>, device <device name>**

---

**Error Type:**

Warning

**Possible Cause:**

A tag references an address using an address type that is not valid for panel module '<subtype>'.

**Solution:**

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

---

**Address <address> is out of range for module <subtype> in slot <slot>, device <device name>**

---

**Error Type:**

Warning

**Possible Cause:**

A tag address references a location that is beyond the range of supported locations for module '<subtype>' in slot '<slot>'.

**Solution:**

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

---

**Address <address> is out of range for panel module <subtype>, device <device name>**

---

**Error Type:**

Warning

**Possible Cause:**

A tag address references a location that is beyond the range of supported locations for panel module '<subtype>'.

**Solution:**

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

---

**Port <port> of the base module returned an error with value = <error value>**

---

**Error Type:**

Warning

**Possible Cause:**

The base module's serial port <port> generated an error during its operation. All tags referencing this slot will be invalidated.

**Solution:**

Check the error list for a detailed description of this error value. If is not included in the list, contact device manufacturer for assistance. Upon a successful block read, this error state will clear and all tags referencing this slot will be validated.

**See Also:**

[Optilogic Device Error Codes](#)

---

**Module <subtype> in slot <slot> returned an error with value = <error value>**

---

**Error Type:**

Warning

**Possible Cause:**

Module '<subtype>' in slot <slot>, generated an error during its operation. All tags referencing this slot will be invalidated.

**Solution:**

Check the error list for a detailed description of this error value. If is not included in the list, contact the device manufacturer for assistance. Upon a successful block read, this error state will clear and all tags referencing this slot will be validated.

**See Also:**

[Optilogic Device Error Codes](#)

---

**Port <port> of module <subtype> in slot <slot> returned an error with value = <error value>**

---

**Error Type:**

Warning

**Possible Cause:**

Serial port <port> of module '<subtype>' in slot <slot>, generated an error during its operation. All tags referencing this slot will be invalidated.

**Solution:**

Check the error list for a detailed description of this error value. If is not included in the list, contact device manufacturer for assistance. Upon a successful block read, this error state will clear and all tags referencing this slot will be validated.

**See Also:**

[Optilogic Device Error Codes](#)

---

**Panel module <subtype> returned an error with value = <error value>**

---

**Error Type:**

Warning

**Possible Cause:**

Panel module '<subtype>', generated an error during its operation. All tags referencing this panel will be invalidated.

**Solution:**

Check the error list for a detailed description of this error value. If is not included in the list, contact device manufacturer for assistance. Upon a successful block read, this error state will clear and all tags referencing this panel will be validated.

**See Also:**

[Optilogic Device Error Codes](#)

---

**Write Errors**

---

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

**Write Errors**

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

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

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

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

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

[Write rejected. Address <address> contains an invalid address type for RTU module, device <device name>](#)

[Write rejected. Address <address> contains an invalid address type for module <subtype>, slot <slot>, device <device name>](#)

[Write rejected. Address <address> contains an invalid address type for panel module <subtype>, device <device name>](#)

Write rejected. Address <address> is out of range for module <subtype> in slot <slot>, device <device name>

Write rejected. Address <address> is out of range for panel module <subtype>, device <device name>

Write failed. Port <port> of the base module returned an error with value = <error value>

Write failed. Module <subtype> in slot <slot> returned an error with value = <error value>

Write failed. Port <port> of module <subtype> in slot <slot> returned an error with value = <error value>

Write failed. Panel module <subtype> returned an error with value = <error value>

### **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 OptiLogic RTU 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.

### **Write 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 rejected. Base module referenced in address <address> on device <device name> does not exist**

---

**Error Type:**

Warning

**Possible Cause:**

A write tag address references an RTU that is currently not supported.

**Solution:**

1. Verify that the address is correct; if it is not, re-enter it in the client application.
2. Contact Technical Support regarding the RTU that is not supported.

### **Write rejected. Slot module referenced in address <address> on device <device name> does not exist**

---

**Error Type:**

Warning

**Possible Cause:**

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

**Solution:**

1. Verify that the address is correct; if it is not, re-enter it in the client application.
2. Verify that the module fits securely into its mating connector on the motherboard.

**Write rejected. Panel module referenced in address <address> on device <device name> does not exist**

---

**Error Type:**

Warning

**Possible Cause:**

A write tag address references a panel that is not connected to the RTU.

**Solution:**

1. Verify that the address is correct; if it is not, re-enter it in the client application.
2. Verify that both ends of the interconnect cable are securely connected.

**Write rejected. Address <address> contains an invalid address type for RTU module, device <device name>**

---

**Error Type:**

Warning

**Possible Cause:**

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

**Solution:**

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

**Write rejected. Address <address> contains an invalid address type for module <subtype>, slot <slot>, device <device name>**

---

**Error Type:**

Warning

**Possible Cause:**

A write tag references an address using an address type that is not valid for module '&lt;subtype&gt;' in slot '&lt;slot&gt;'.

**Solution:**

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

**Write rejected. Address <address> contains an invalid address type for panel module <subtype>, device <device name>**

---

**Error Type:**

Warning

**Possible Cause:**

A write tag references an address using an address type that is not valid for panel module '&lt;subtype&gt;'.

**Solution:**

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

**Write rejected. Address <address> is out of range for module <subtype> in slot <slot>, device <device name>**

---

**Error Type:**

Warning

**Possible Cause:**

A write tag address references a location that is beyond the range of supported locations for module '&lt;subtype&gt;' in slot '&lt;slot&gt;'.

**Solution:**

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



---

**Write rejected. Address <address> is out of range for panel module <sub-type>, device <device name>**

---

**Error Type:**

Warning

**Possible Cause:**

A write tag address references a location that is beyond the range of supported locations for panel module '<sub-type>'.  
</p></div>

**Solution:**

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

---

**Write failed. Port <port> of the base module returned an error with value = <error value>**

---

**Error Type:**

Warning

**Possible Cause:**

The base module's serial port <port> generated an error during a write operation. Write failed.

**Solution:**

Check the error list for a detailed description of this error value. If is not included in the list, contact device manufacturer for assistance. Upon a successful block read, this error state will clear and all tags referencing this slot will be validated.

**See Also:**

[Optilogic Device Error Codes](#)

---

**Write failed. Module <subtype> in slot <slot> returned an error with value = <error value>**

---

**Error Type:**

Warning

**Possible Cause:**

Module '<subtype>' in slot <slot>, generated an error during a write operation. Write failed.

**Solution:**

Check the error list for a detailed description of this error value. If is not included in the list, contact device manufacturer for assistance. Upon a successful block read, this error state will clear and all tags referencing this slot will be validated.

**See Also:**

[Optilogic Device Error Codes](#)

---

**Write failed. Port <port> of module <subtype> in slot <slot> returned an error with value = <error value>**

---

**Error Type:**

Warning

**Possible Cause:**

Serial port <port> of module '<subtype>' in slot '<slot>' generated an error during a write operation. Write failed.

**Solution:**

Check the error list for a detailed description of this error value. If is not included in the list, contact device manufacturer for assistance. Upon a successful block read, this error state will clear and all tags referencing this slot will be validated.

**See Also:**

[Optilogic Device Error Codes](#)

---

**Write failed. Panel module <subtype> returned an error with value = <error value>**

---

**Error Type:**

Warning

**Possible Cause:**

Panel module '<subtype>', generated an error during a write operation. Write failed.

**Solution:**

Check the error list for a detailed description of this error value. If is not included in the list, contact device manufacturer for assistance. Upon a successful block read, this error state will clear and all tags referencing this panel will be validated.

**See Also:**

[OptiLogic Device Error Codes](#)

# Index

## 1

16 Channel Analog Input Module .....	27
16 Channel Analog Output Module .....	29
16 Digital Input Module .....	12
16 Digital Output Module .....	19

## 2

2 Channel Analog Input Module .....	25
2 Channel Analog Output Module .....	27
2 Channel High Speed Counter .....	35, 70
2 Port Serial .....	43, 81
24 Digital Input Module .....	13
24 Digital Output Module .....	21

## 3

32 Digital Input Module .....	13
32 Digital Output .....	23

## 4

4 Channel Analog Input Module .....	26
4 Channel Analog Output Module .....	28, 79
4 Digital Input Module .....	11, 68
4 Digital Output Module .....	14, 59

## 8

8 Channel Analog Input .....	26, 80
8 Channel Analog Output Module .....	29
8 Digital Input Module .....	11, 68-70

8 Digital Output ..... 16, 61, 63, 66

**A**

Address <address> contains an invalid address type for module <subtype>, slot <slot>, device <device name> ..... 108

Address <address> contains an invalid address type for panel module, <subtype> device <device name> ..... 109

Address <address> contains an invalid address type for RTU module, device <device name> ..... 108

Address <address> is out of range for module <subtype> in slot <slot>, device <device name> ..... 109

Address <address> is out of range for panel module <subtype>, device <device name> ..... 109

Address <address> is out of range for the specified device or register ..... 105

Address Descriptions ..... 10

Array size is out of range for address <address> ..... 106

Array support is not available for the specified address: <address> ..... 106

**B**

Base module referenced in address <address> on device <device name> does not exist ... 108

Base Serial Port ..... 38, 97

Base with type=<type> Major ver.=<major> Minor ver.=<minor> is currently not supported. Contact technical support ..... 106

Boolean ..... 9

**D**

Data Type ..... 9

Data type <type> is not valid for device address <address> ..... 106

Data Types Description ..... 9

Device '<device name>' is not responding ..... 105

Device address <address> contains a syntax error ..... 105

Device address <address> is not supported by model <model name> ..... 105

Device address <address> is Read Only ..... 106

Device ID ..... 6

Device Setup ..... 6

Digital I/O Bit Mapping ..... 10

Driver Error Messages.....	104
Driver Warning Messages.....	104
Dual RS232 Port Module.....	42, 81

## E

Error Descriptions.....	103
-------------------------	-----

## F

Frame received from device <device name> contains errors.....	107
---	-----

## H

High Speed Counter Module.....	30, 73
--------------------------------	--------

## I

IP Address.....	6
-----------------	---

## M

Memory allocation error.....	104
Module <subtype> in slot <slot> returned an error with value =<error value >.....	109
Module in slot <slot> w/ type=<type> subtype=<subtype> is currently not supported. Contact Technical Support.....	107

## N

Network.....	6
--------------	---

**O**

OL3406 Operator Panel.....	47, 85
OL3420 Operator Panel.....	50, 89
OL3440 Operator Panel.....	54, 92
OL3850 Operator Panel.....	54, 93
OptiLogic Device Errors.....	104
Optimizing Your Optimation OptiLogic Communications.....	8
Overview.....	5

**P**

Panel module <subtype> returned an error with value=<error value>.....	110
Panel module referenced in address <address> on device <device name> does not exist ..	108
Panel with type=<type> subtype=<subtype> is currently not supported. Contact Technical Support ..	107
Port <port> of module <subtype> in slot <slot> returned an error with value=<error value>.....	110
Port <port> of the base module returned an error with value = <error value>.....	109

**R**

Read Errors.....	107
------------------	-----

**S**

Short.....	9
Slot module referenced in address <address> on device <device name> does not exist ..	108
Socket.....	104

**U**

Unable to write to <address> on device <device name>.....	111
---	-----

## W

Winsock .....	104
Winsock initialization failed (OS Error = n) .....	104
Winsock V1.1 or higher must be installed to use the OptiLogic device driver .....	104
Word .....	9
Write Errors .....	110
Write failed. Frame received from device <device name> contains errors .....	111
Write failed. Module <subtype> in slot <slot> returned an error with value=<error value> ..	113
Write failed. Panel module <subtype> returned an error with value=<error value> .....	114
Write failed. Port <port> of module <subtype> in slot <slot> returned an error with value=<error value> .....	113
Write failed. Port <port> of the base module returned an error with value=<error value> ..	113
Write rejected. Address <address> contains an invalid address type for panel module <subtype> device <device name> .....	112
Write rejected. Address <address> contains an invalid address type for RTU module device <device name> .....	112
Write rejected. Address <address> contains an invalid address type for module <sub- type>, slot <slot>_ device name .....	112
Write rejected. Address <address> is out of range for module <subtype> in slot <slot> device <device name> .....	112
Write rejected. Address <address> is out of range for panel module <subtype> device <device name> .....	113
Write rejected. Base module referenced in address <address> on device <device name> does not exist .....	111
Write rejected. Panel module referenced in address <address> on device <device name> does not exist .....	112
Write rejected. Slot module referenced in address <address> on device <device name> does not exist .....	111