Appendix 12
Generic Serial, Generic IP, and Generic Server

This appendix describes support for interfaces that implement a protocol and action that is not directly supported by HCA.

Generic interfaces are supported only to the extent that HCA can send to them and receive from them and, optionally, decode them only to the extent of breaking receptions into individual messages. Data sent must be assembled by your own implementation in one or more Visual Programs. Data received must be processed by your own implementation in one or more Visual Programs.

Generic serial and IP (network) interfaces are configured and used in approximately the same manner. Because so much is common between them both are covered in this appendix.

Not all interfaces that are attached to the computer using a serial port – real or virtual – are a *Generic Serial Interface* as covered in this appendix. For example, the SmartHome Serial Insteon interface isn't a *Generic Serial Interface* because HCA includes an implementation of its protocol.

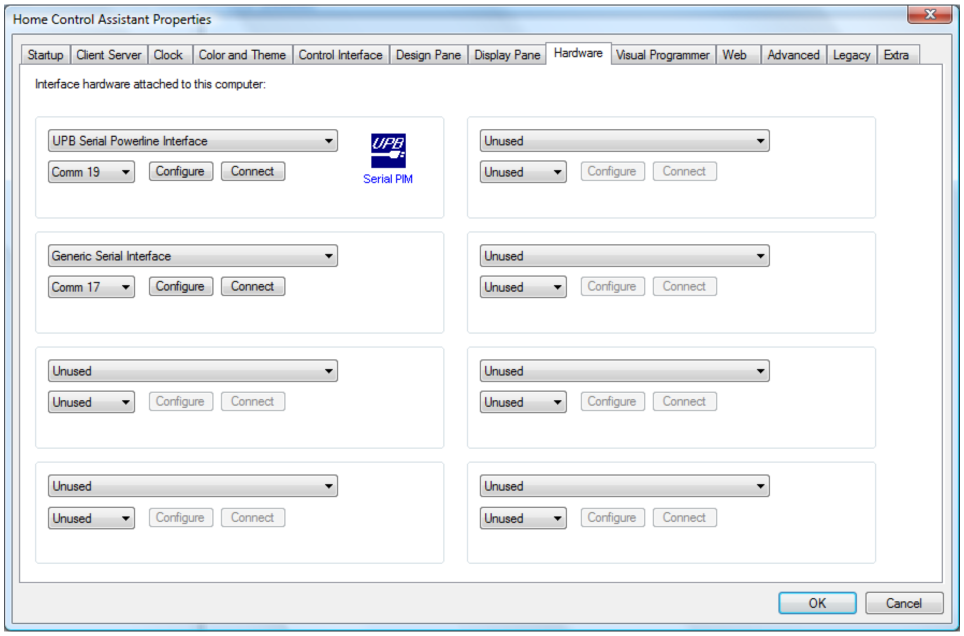
Not all interfaces that are accessed using a network connection are Generic IP Interfaces as covered in this appendix. For example, the Global Cache IR Interface has a network address and communication is made using TCP but it isn't a *Generic IP Interface* because HCA includes an implementation of its protocol.

This appendix covers these topics:

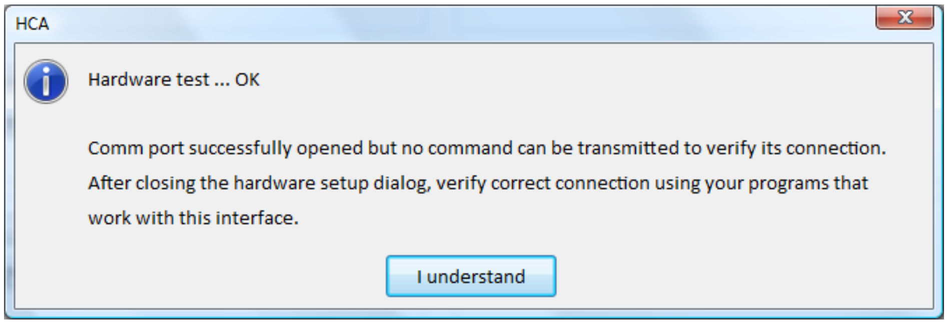
- Configuring a generic serial interface
- Configuring generic IP interfaces and Generic Server interfaces
  - Client interfaces
  - Server interfaces
- Visual program Port I/O element
- Port Reception Triggers
- Working with Generic Interfaces

Configuring a serial interface

A generic serial interface is configured from *HCA Options* on the *Interfaces* tab.



Choose "Generic Serial Interface" as the interface type, select the serial port to use, and press the *Connect* button. If the serial port can be opened a popup message appears:

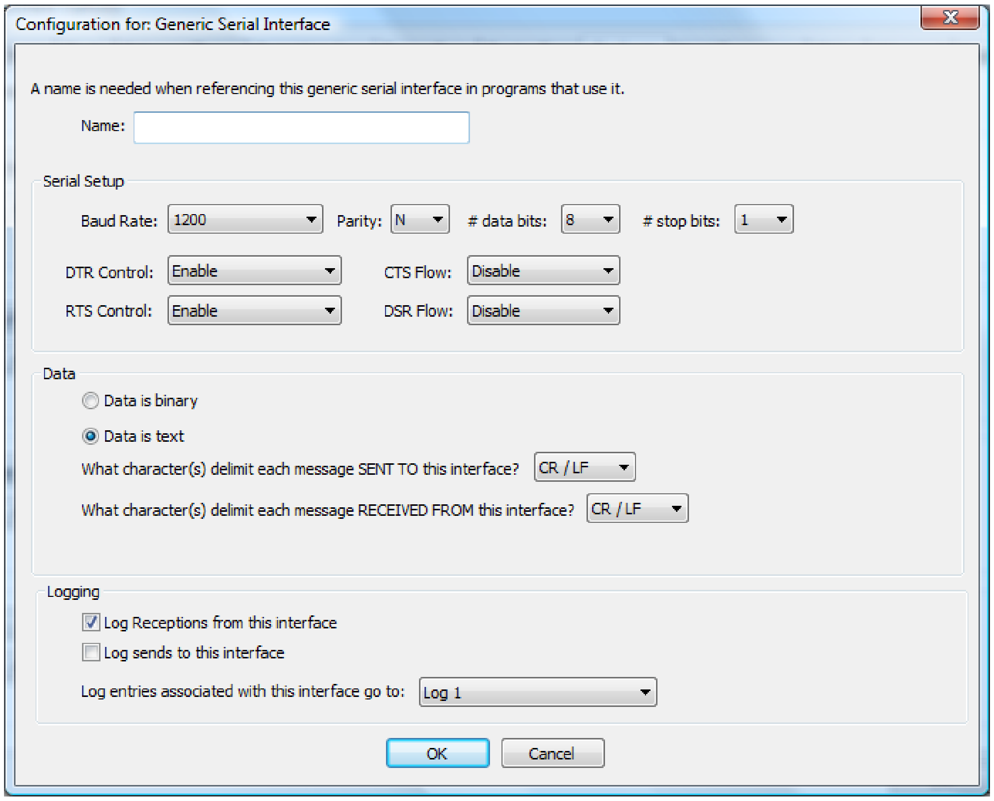

HCA

Hardware test ... OK

Comm port successfully opened but no command can be transmitted to verify its connection. After closing the hardware setup dialog, verify correct connection using your programs that work with this interface.

As the dialog says, the only test performed is that the serial port can be opened. There doesn't even have to be an actual interface connected to that serial port for the test to pass.

Serial communication has a number of settings: Baud rate, flow control, etc. These settings for this interface can be configured. Press the *Configure* button to open the settings dialog.



A name is needed when referencing this generic serial interface in programs that use it.

Name:

Serial Setup

Baud Rate: 1200 Parity: N # data bits: 8 # stop bits: 1

DTR Control: Enable CTS Flow: Disable

RTS Control: Enable DSR Flow: Disable

Data

○ Data is binary

● Data is text

What character(s) delimit each message SENT TO this interface? CR / LF

What character(s) delimit each message RECEIVED FROM this interface? CR / LF

Logging

☑ Log Receptions from this interface

☐ Log sends to this interface

Log entries associated with this interface go to: Log 1

Since you can have more than one serial interface in use simultaneously, by giving each a name you can use that name to identify which interface you want to work with in a program.

The sections of this dialog are:

Serial Setup

Baud rate, parity, # of date bits, # of stops bits
These are the standard serial port settings and they depend upon the interface you are using.

DTR Control, RTS Control, CTS Flow, DSR Flow
These are the serial flow control settings and the choices depending upon the connected interface.

Text Data

Each message sent to and from the serial interface is decoded to a very minimal degree. This decoding needs to know how messages are separated from each other. In this way when you use the Visual Program *Port IO* element to receive a message, the decode mechanism knows when a message has been completely received. It is completely received when the message terminator(s) are received.

When a message is transmitted a terminator can be added. The *Sent To* terminator can be the same or different for received messages.

The choices for the send and receive terminators are:

- CR – ascii hex 0d
- LF – ascii hex 0a
- CR followed by LF
- LF followed by CR

- None
- Other

In the case of the CR-LF and LF-CR then two characters are the message terminator. In the case of *Other* then you can enter the character to use as the terminator. Entry is made as a hex number of the ascii character code so non-printable characters can be used.

If *None* is used as the *Sent To* terminator then no terminator characters are automatically added to each message sent. With the *Sent To* terminator set to *none* then programs that use the Port IO element must included the necessary terminating characters in each send.

If *None* is used as the receive terminator then no decoding of receptions into separate messages happens. With the receive terminator set to *None*, whenever the *Port IO* element configured for a receive executes, then anything currently in the receiver buffer is what is read. If the buffer is empty then the element immediately times out.

Binary Data

Each message sent to and from the serial interface is decoded to a very minimal degree. This decoding needs to know how messages are separated from each other. In this way when you use the Visual Program *Port IO* element to receive a message, the decode mechanism knows when a message has been completely received. Unlike text data, binary data doesn't have a terminating character. HCA determines when a complete message is received based upon your selection in the message type dropdown. There are several choices and which you use depends upon the hardware being interfaced with. As an example, suppose the data being sent was the 3 bytes 0x40 0x00 0xef. These are the options and how the message data is expected.

- Fixed length
  0x40 0x00 0xef
- Byte length prefix. Record length not including length byte
  0x030x40 0x00 0xef
- Byte length prefix. Record length including the length byte
  0x04 0x40 0x00 0xef
- Word (high – low) length prefix. Record length not including the length word
  0x00 0x03 0x40 0x00 0xef
- Word (low –high) length prefix. Record length not including the length word
  0x03 0x00 0x40 0x00 0xef
- Word (high –low) length prefix. Record length including the length word
  0x00 0x04 0x40 0x00 0xef
- Word (low - high) length prefix. Record length not including the length word
  0x04 0x00 0x40 0x00 0xef

When programs receive the data using the Port I/O element the data is in a text string. Continuing the above example the data would be in the string "4000ef". Data is supplied in the same manner when sending data to the serial interface.

If the "binary send formatted same as reception type" option is enabled, then if one of the length prefix options is selected, the length prefix byte(s) are prefixed to the data being sent.

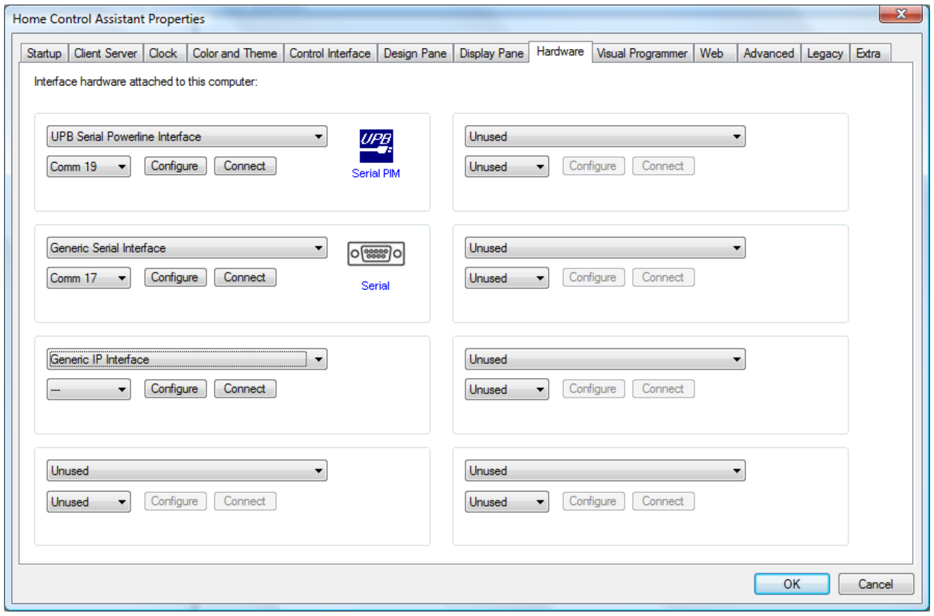
Logging

As with all interfaces you can select what is logged and the log that is used. Select the log to use and if sends or receives or both are logged.

The checkbox to configure for printable characters is used in logging of the sends and receives. If the data is printable characters then it can be logged as text otherwise a hex representation of the received bytes is used.

Configuring an IP interface

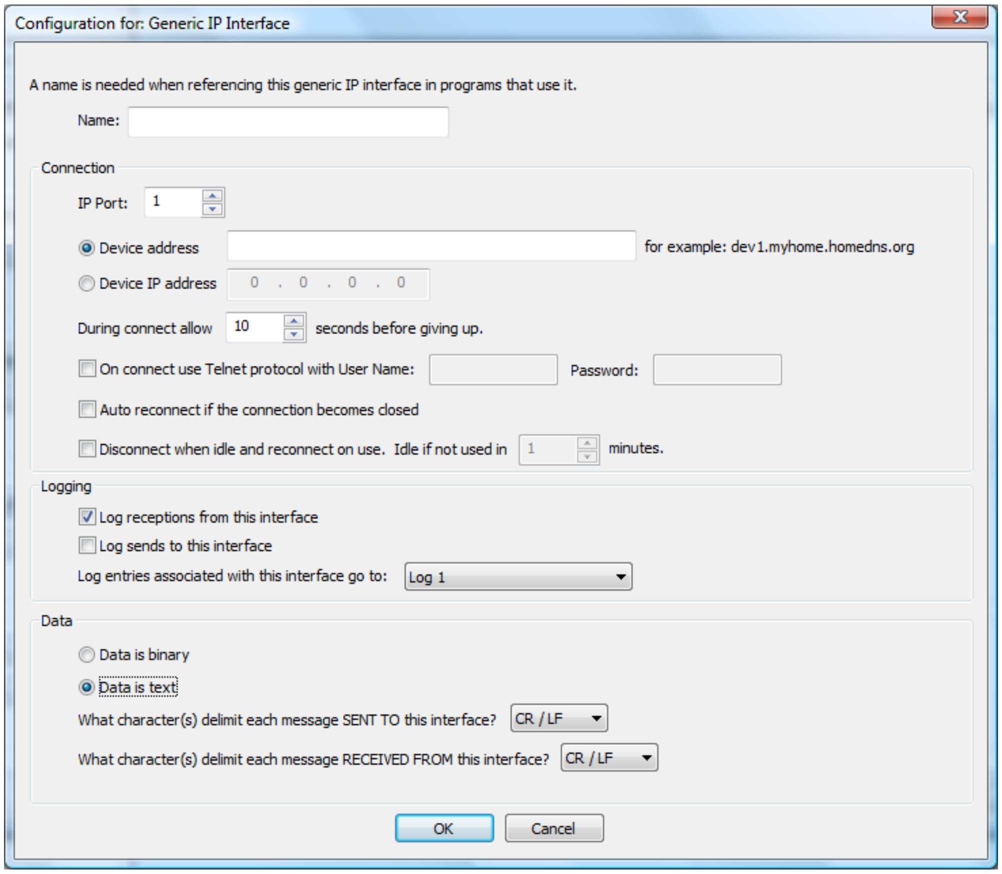
An IP interface is connected from *HCA Options* on the *Interfaces* tab.



Interface hardware attached to this computer:

UPB Serial Powerline Interface — Comm 19
Generic Serial Interface — Comm 17
Generic IP Interface — ---
Unused — Unused

Like the serial interface, before you can test the connection it is necessary to configure it. Press the *Configure* button.


Configuration for: Generic IP Interface

A name is needed when referencing this generic IP interface in programs that use it.

Name:

Connection

IP Port: 1

[✓ Device address] for example: dev1.myhome.homedns.org

Device IP address 0 . 0 . 0 . 0

During connect allow 10 seconds before giving up.

On connect use Telnet protocol with User Name: Password:

Auto reconnect if the connection becomes closed

Disconnect when idle and reconnect on use. Idle if not used in 1 minutes.

Logging

[✓ Log receptions from this interface]

Log sends to this interface

Log entries associated with this interface go to: Log 1

Data

Data is binary

[✓ Data is text]

What character(s) delimit each message SENT TO this interface? CR / LF

What character(s) delimit each message RECEIVED FROM this interface? CR / LF

In this dialog you configure both the connection and the format of the data being transmitted. The data type – binary or text – and all the options for them are the same as in the generic serial interface.

Since you can have more than one IP interface in use simultaneously, by giving each a name, you can use that name to identify which interface you want to send to or receive from in a program.

There are two ways to configure a Generic IP Interface: As a client or server. The next section explains the differences.

Client or Server

Before discussing the details of the connection it is necessary to determine if an interface is used for a client connection or as a server.

When HCA opens a network interface, HCA can operate as a *client* of that interface or HCA can operate as a *server*.

When creating a connection to an interface as a client, what you are saying is that there is some physical interface someplace on the network. Using HCA programs you send messages to this interface and receive responses back.

When configuring an interface as a *server* what you are saying is that HCA **is** the interface. Other programs (Windows programs, browsers, mobile applications) open the IP address of the computer that HCA is running on with the port number specified in the interface configuration. Those programs then send requests to HCA and receive responses back. HCA programs that you create receive those messages, process them, and reply with responses.

Why would you want to configure an interface to act as a server? Using a server you can provide an interface for mobile apps, windows programs, and browser addins to send messages into HCA for action. What are those messages? That's up to you! You get to design whatever messages are needed to accomplish whatever your task is.

Configuring an IP Interface

The parameters to configure an IP interface:

Address

When configuring the interface as a client then you need to specify the IP address of the device you are communicating with. That interface is located by a 4-part network address given in the form A.B.C.D. You can supply the address in that format or, if a DNS entry exists for the interface, then you can use its name. Upon connection, a lookup is done on the name to translate it into the 4-part network address.

In addition to a network address, a port number is needed as a network interface may open different ports for different purposes.

Timeouts

In any communication you always have to handle the case where the communication fails. The *During connect allow* timeout setting is used during connection to the interface. If after the given number of seconds a connection is not established, then HCA gives up and shows an error.

Auto Disconnect

Some interfaces allow only a single connection at a time. This can be a problem if HCA opens the connection and leaves it open forever. Doing that may prevents other users of that interface from connecting to it.

If the *Disconnect* option is enabled, then HCA connects to the interface when it needs to use it, and then leaves it open for the specified number of minutes and if not used again during that time then it disconnects. The next time it needs to use the interface then it reconnects.

If you use this option then you will not receive *Asynchronous* messages from the interface. Here is what that means: Suppose you are connecting to an interface that controls a hard-wired lighting system. You can send commands to it to turn on and off the lights. You send a message saying "Lights on" and it replies back with "OK". That a *synchronous* message protocol. You send something and it replies. The message from the interface (its reply) is always preceded by a message sent to it (your request).

But also suppose that the interface can notice when someone pushes a button to turn the lights on. It could send to HCA a message to report what occurred. This would be an asynchronous message. You didn't ask it to do anything; it just sends you a message. The message you received (the report) isn't preceded by a message you sent to it. That is why this is an a*synchronous* message.

If the interface is not connected when the interface has something to report in this way, it will not get received by HCA since HCA doesn't have an open connection to it.

If you want to receive these kinds of messages – reports for actions that HCA didn't initiate – then you can't use the disconnect option.

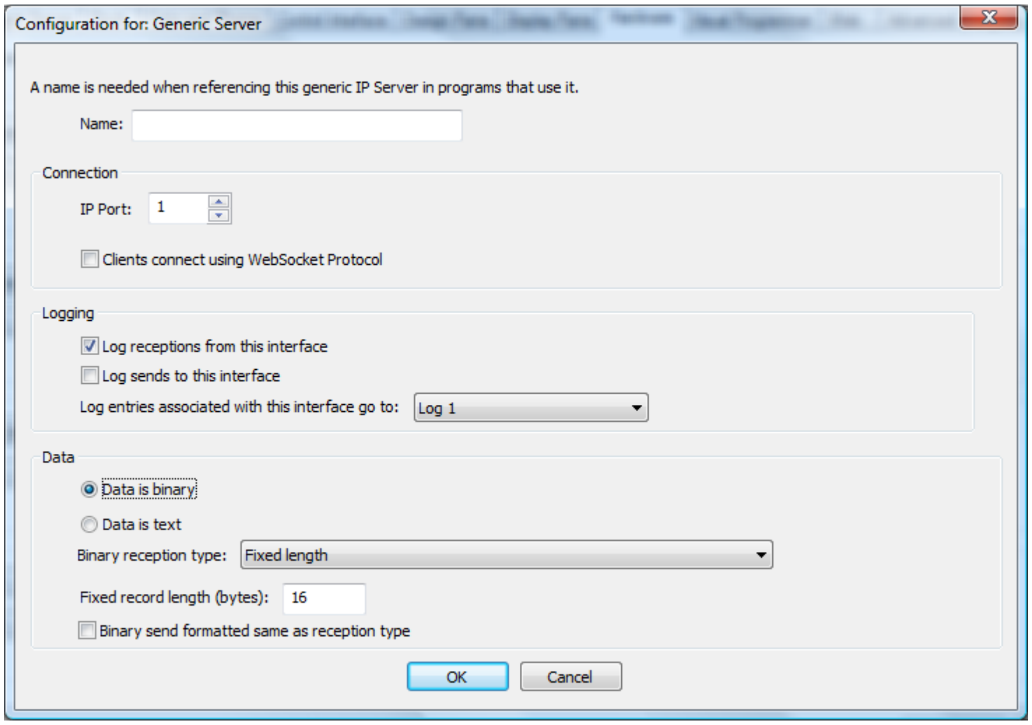
Telnet

If the interface you are using operates as a Telnet server and if you want to connect to it that way, then tick the *Use Telnet protocol* checkbox and enter the user name and password needed. Each time HCA connects to the interface it automatically performs a telnet login using that user name and password.

Tip: Make sure you use the appropriate port. Telnet is often on port 23.

Configuring a Generic Server

To create a generic server, on the hardware connection tab of HCA Options, select *Generic Server* and then press the *Configure* button.



Configuration for: Generic Server

A name is needed when referencing this generic IP Server in programs that use it.

Name:

Connection

IP Port: 1

☐ Clients connect using WebSocket Protocol

Logging

☑ Log receptions from this interface

☐ Log sends to this interface

Log entries associated with this interface go to: Log 1

Data

● Data is binary

○ Data is text

Binary reception type: Fixed length

Fixed record length (bytes): 16

☐ Binary send formatted same as reception type

The main parameter is the port number. This port number must **not** be the port number the HCA Server uses and HCA clients – and mobile applications – connect to.

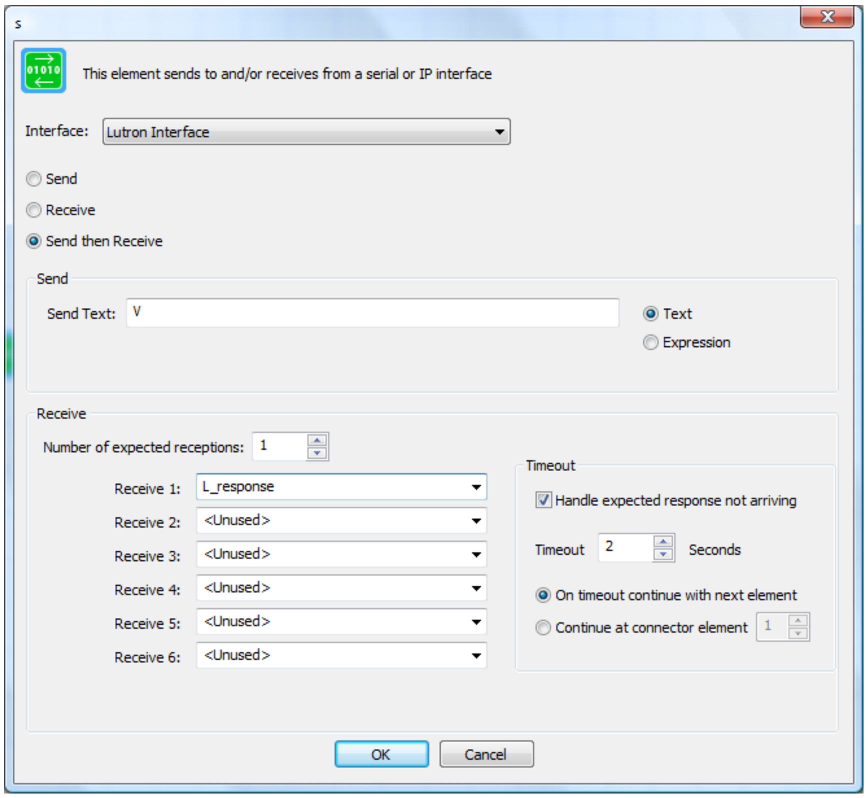
Like the Generic IP Interface and the Generic Serial interface, you can specify the data and hopw the data is handled – binary or text. If you are making the connection to the generic server using a WebSocket then the WebSocket protocol frames and un-frames the messages so the delimiters are not needed.

Note: There is a technical note more completely describing the use of a Generic Server with an example. Look for the *User Applications* note in the technical notes area of the support web site.

Send and Receive

As described in the introduction to this appendix, HCA manages sending to and receiving to the interface. But the contents of the data sent and received is opaque to HCA – all decoding has to be done in programs you create.

Sending and receiving from the serial or IP interfaces uses the Visual Program *Port I/O* element. This element, depending upon its configuration, can perform three different actions: Send, Receive, or Send and then Receive.


s

This element sends to and/or receives from a serial or IP interface

Interface: Lutron Interface

Send
Receive
[✓ Send then Receive]

Send

Send Text: V [✓ Text] Expression

Receive

Number of expected receptions: 1

Receive 1: L_response
Receive 2: <Unused>
Receive 3: <Unused>
Receive 4: <Unused>
Receive 5: <Unused>
Receive 6: <Unused>

Timeout

[✓ Handle expected response not arriving]

Timeout 2 Seconds

[✓ On timeout continue with next element]
Continue at connector element 1

Select from the interface dropdown the name of the interface to use and then choose the option for send, receive, or send then receive.

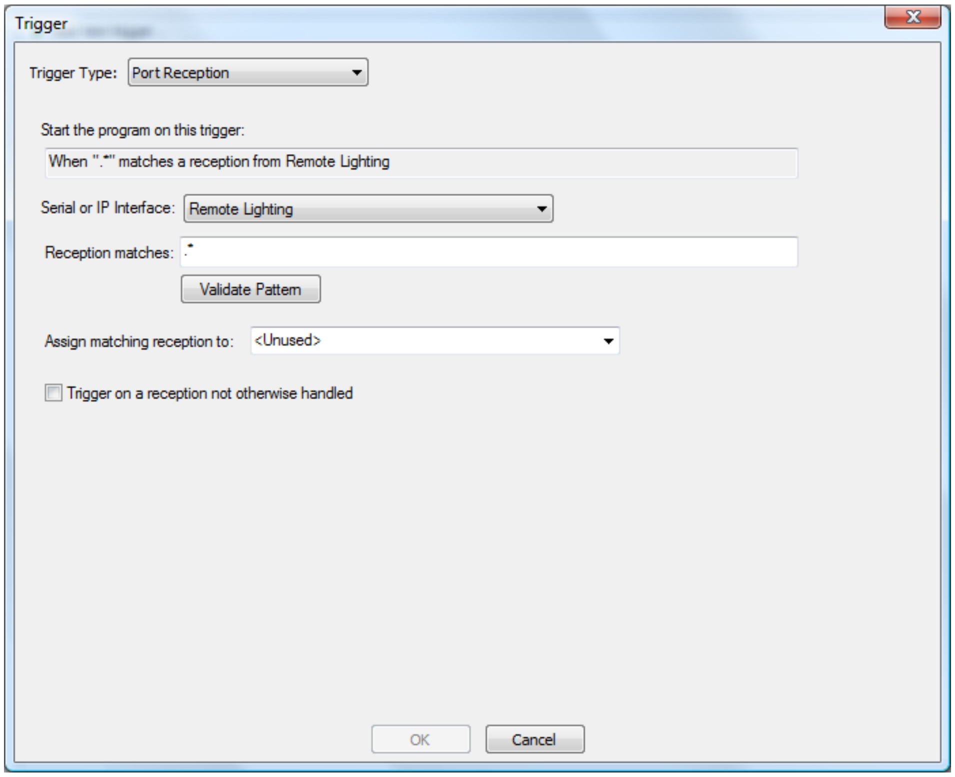
If sending enter the text to send into the *Send Text* box. This can be the actual text or the result of an expression evaluation if the *Expression* option is used.

If receiving enter the number of expected responses and then, for each response, select the flag that reception is assigned to. You can select an existing flag or type in the name of a new flag that is created.

As with any communication, you can handle the case of what happens if an expected reception doesn't arrive. The *Timeout* group contains the options for this. Enter the maximum number of seconds to wait for the expected receptions. If that timeout limit is reached then execution can either continue at the next element or at an connector element with the selected number.

Triggers

Asynchronous messages received from an interface can be used as a trigger for a program. To create a trigger of this type, select as the trigger type *Port Reception*



Trigger Type: Port Reception

Start the program on this trigger:
When ".*" matches a reception from Remote Lighting

Serial or IP Interface: Remote Lighting

Reception matches: .*

Assign matching reception to: <Unused>

☐ Trigger on a reception not otherwise handled

There are several parts to a Port Reception trigger and these are covered in the next sections.

As you review the next sections, remember that if you are working with binary data the text is rendered into characters. For example, a reception of the bytes 0x64 0x00 0xef assigns the string "6400ef" to the chosen variable.

Pattern

Unlike, for example, Insteon triggers where the reception from an Insteon interface is decoded into a source device and command, and Insteon triggers specify that source and command, the Port Reception trigger is based only on the text of the reception itself.

A Port Reception trigger contains a pattern that is matched against the received text. If the pattern matches the received text then the program is triggered. If the reception doesn’t match the pattern then the program is not triggered.

The pattern is specified as a *Regular Expression*. A regular expression is a sequence of characters that forms a search pattern. The full description of a regular expression is beyond the scope of this User Guide but here are the fundamentals.

A regular expression is composed of a series of characters and metacharacters. Characters that are not metacharacters match the corresponding changes in the text. Meta characters describe one or more possible characters to match in the text. The meta characters are:

Character	Use
. (dot)	Matches any single character. For example, a.c matches "abc", or "axc" or "a6c" etc.
[ ]	A bracket expression. Matches a single character that is contained within the brackets. For example, [abc] matches "a", "b", or "c". [a-z] specifies a range which matches any lowercase letter from "a" to "z". These forms can be mixed: [abcx-z] matches "a", "b", "c", "x", "y", or "z", as does [a-cx-z].

[^]	Matches a single character that is not contained within the brackets. For example, [^abc] matches any character other than "a", "b", or "c". [^a-z] matches any single character that is not a lowercase letter from "a" to "z".
^	Matches the starting position within the string
$	Matches the ending position of the string
*	Matches the preceding element zero or more times. For example, ab*c matches "ac", "abc", "abbbc", etc. [xyz]* matches "", "x", "y", "z", "zx", "zyx", "xyzzy", and so on.
{m, n}	Matches the preceding element at least m and not more than n times. For example, a{3,5} matches only "aaa", "aaaa", and "aaaaa". This is not found in a few older instances of regular expressions
\	The following character is a character and not a metacharacter

Here are some examples:

- `.*` matches any text
- `.at` matches any three-character string ending with "at", including "hat", "cat", and "bat".
- `[hc]at` matches "hat" and "cat".
- `[^b]at` matches all strings matched by .at except "bat".
- `[^hc]at` matches all strings matched by .at other than "hat" and "cat".
- `^[hc]at` matches "hat" and "cat", but only at the beginning of the string or line.
- `[hc]at$` matches "hat" and "cat", but only at the end of the string or line.
- `\[.\]` matches any single character surrounded by "[" and "]" since the brackets are escaped, for example: "[a]" and "[b]".

Since it can be complex to compose a regular expression, in the dialog is a *Validate* button that checks the syntax of the regular expression. You must use the *Validate* button to check the pattern before creating the trigger.

However, just because the syntax is correct that doesn't mean that the pattern matches strings as you expect. Only testing shows that.

Tip: Use your favorite search engine or Wikipedia to find the full syntax of regular expressions.

Flag assignment

Port Reception triggers are also different than other triggers in an additional aspect. The text of the reception can be assigned to a flag for subsequent decoding in the triggered program.

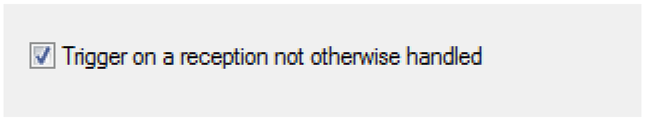
Select the name of an existing flag or type in the name of a new flag. If the program is triggered – when the pattern matches the reception – the received text is assigned to the flag before the program starts. This lets you do further decoding of the reception in the program that is started.

This is an optional feature of the trigger. If you don't want to use this, set the flag as *<Unused>*

Trigger of last resort

You can also create a trigger that starts a program if no trigger on any program matches the reception. Of course, only one program can be designated with this trigger.

To enable this option tick the *Trigger on a reception not otherwise handled* checkbox.

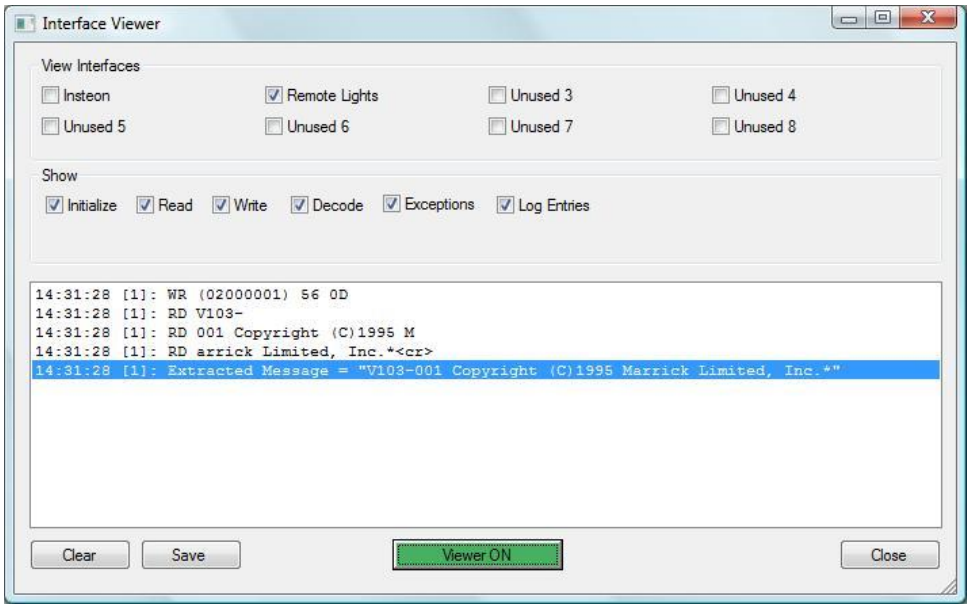


[✓] Trigger on a reception not otherwise handled

Working with Serial and IP Interfaces

It can be complex working with these Generic interfaces since all the encoding of messages you send and decoding of messages received is implemented in your programs.

Open the *Interface Viewer* from the *Interfaces* ribbon category. This viewer shows the messages sent to and received from one or more interfaces. This gives you a window into the communications at the lowest level.



```
14:31:28 [1]: WR (02000001) 56 0D
14:31:28 [1]: RD V103-
14:31:28 [1]: RD 001 Copyright (C)1995 M
14:31:28 [1]: RD arrick Limited, Inc.*<cr>
14:31:28 [1]: Extracted Message = "V103-001 Copyright (C)1995 Marrick Limited, Inc.*"
```

The names of the eight interfaces that can be configured are the first set of checkboxes. Tick the ones that you want to see data from. Interfaces not configured are listed as *Unused.*

The *Show* box determines what sort of information you want to view. Tick or clear the boxes for the type of data you want to see.

You can turn the viewer on or off with the button at the bottom of the dialog. When green the viewer is enabled and when red it isn't.

In the list messages received from the interface are prefixed with RD, message sent to the interface are prefixed with WR, and if the message is decoded – messages from some interfaces are decoded but messages from the Generic Interfaces are not – they are prefixed with DC. Enclosed in []'s is the number of the interface the line is for. The first interface is numbered 0 and the last is numbered 7.

This interface is at a very low level so you may see, because of the nature of network and serial reads, a single reception broken across in multiple RD lines.

You can clear the list with the *Clear* button and save the list into a text file with the *Save* button.