

# BitWise controls

TM

## BC4/BCX Command Protocol



Revision 1.6 – April 7, 2010

Copyright © 2009 – 2010 by BitWise Controls, LLC – All Rights Reserved

[www.bitwisecontrols.com](http://www.bitwisecontrols.com)

Phone: 866-932-2BWC

## Contents

Contents.....	2
Introduction .....	4
Attention .....	4
Overview .....	4
Command Reference Conventions and Notes.....	4
Packet Structure .....	4
Format.....	4
Preamble.....	5
Get/Set.....	5
Type.....	5
Parameters & Values .....	5
Command Reference.....	6
Discovery & Location .....	6
Discover.....	6
Locate.....	7
Network Settings & Configuration .....	8
IPconfig.....	8
PortConfig .....	9
TCPTimeout.....	9
Keepalive.....	10
TCPstatus.....	11
TCPclose .....	11
Heartbeat .....	11
UDPsendto .....	12
UDPsend.....	13
RS (RemoteSend) .....	13
TCP Client .....	14
TCPCopen .....	14
TCPCclose .....	14
TCPCstatus .....	14
TCPCsendHex .....	15
TCPCsendStr .....	15
Temperature Sensor .....	16
Temp .....	16
TempConfig.....	16
Serial Ports.....	17
SerialConfig .....	17
Send232Hex .....	18

## BitWise Controls BC4 / BCX Command Protocol

---

Send232str .....	18
Send232termstr .....	18
Baud .....	19
TCPSclose .....	20
TCPSstatus.....	20
<b>GPIO</b> .....	<b>20</b>
IOconfig.....	20
Input.....	21
AD.....	22
Output.....	23
Count.....	23
Frequency .....	24
<b>Relays</b> .....	<b>25</b>
Relay.....	25
<b>IR (Infrared)</b> .....	<b>26</b>
SendIR .....	26
SendIR (continuous).....	27
SendLearned .....	28
SendLearned (continuous).....	28
Reset .....	29
Cancel.....	29
GetKeys .....	30
Learn .....	30
<b>Variables</b> .....	<b>31</b>
State .....	31
<b>Macros</b> .....	<b>32</b>
Execute Macro by Number .....	32
Execute Macro by Name.....	32
Get Macro Name.....	33
<b>Misc</b> .....	<b>33</b>
Reset .....	33
Heartbeat .....	33
DelayedCmd.....	34
<b>Resources</b> .....	<b>35</b>
<b>Contact Information</b> .....	<b>35</b>

# Introduction

## Attention

This document is applicable for BC4 firmware revisions beginning with v1.01. Earlier firmware revisions may not include all commands listed in this document.

## Overview

The BitWise Controls Control Protocol allows TCP and UDP clients complete access to all of the BC4's internal settings, as well as all configuration and usage commands for the BCX-series expansion modules. This document is intended to be a complete reference for all commands, responses, and status reporting. Note that some commands may not work with all BCX expansion modules, depending on the available device connections present on a given module.

## Command Reference Conventions and Notes

Unless otherwise noted, quotation marks ( " " ) are not intended to be a part of the referenced command. The text enclosed within quotes is the actual text that should be used in a valid packet.

Unless otherwise noted, the following brackets ( < > ) are not intended to be a part of the referenced command. They are intended to indicate that the value to be sent/received as part of a valid packet.

Unless otherwise noted, parentheses ' ( ) ' are not intended to be a part of the referenced command. They are intended to indicate that the elements within the parenthesis are only included if the data packet is a "set" packet.

Each element in a valid data packet will be delimited by the ":" character.

### **IMPORTANT:**

**ALL TCP PACKETS SENT MUST BE TERMINATED WITH A CARRIAGE RETURN!!!**

This allows control and configuration of the BC4 through a standard telnet session.

## Packet Structure

### Format

Preamble:Get/Set(not present in all packet Types):Type:<additional Parameters or Values>:

### Preamble

A valid BC4 data packet consists of a number of elements, each being separated by the “:” character. The first element in any given packet is the Preamble. There are four basic types of data packets used in all BC4 communications, and each type has a different Preamble.

- “bwc” indicates that the packet is a command being sent to the BC4
- “bwr” indicates that the packet is a reply being sent from the BC4 to the host, generally in direct response to a corresponding “bwc” packet.
- “bws” indicates that the packet is a status update being sent from the BC4 to the host, usually in response to a change in state or value of a particular BCX connector.
- “NAK” indicates that the last command packet received was invalid for some reason. The next elements in a NAK packet will be the last packet received.

### Get/Set

The next element, after the header, is the Get/Set element. Note that this element is not present in every packet type.

“get” indicates that the host is requesting the value or state of a given setting or device.

“set” indicates that the host is changing the value or state of a given setting or device.

### Type

The next element, after the Get/Set element (if present) is the Type element. This element indicates the actual type of data packet in use. There are many valid Types. For the entire listing of available command types and their usage, refer to the Command Reference section of this document.

### Parameters & Values

The next set of elements, after the Type element, will be one or more parameter or value elements, used to contain the applicable data for the given packet Type. Each packet Type has its own set of parameters necessary to make up a valid Packet. Refer to the Command Reference section of this document to see what parameter and value elements are used for each packet Type.

## Command Reference

This section will provide a detailed listing of all valid command, reply, and status packet structures, with examples, broken down into groups by function.

### Discovery & Location

Discovery packets are sent and received on UDP port 30303. This is the only fixed port setting on a BC4, and is set that way to ensure that any given BC4 can be found on a network without having to know its IP address. To find BC4s on a network, you should BROADCAST (send to IP address 255.255.255.255) a discovery packet on UDP port 30303. Each BC4 on the network will then broadcast a discovery status update on UDP port 30303.

#### Discover

**Purpose:**

Used to find any BC4s on the network

**Format:**

“bwc:discover:”

**Status Update:**

“bws:discover:<NAME>:<MAC>:<IP>:<TCP PORT>:<UDP IN PORT>:  
<UDP OUT PORT>:<FIRMWARE V>:<BUILD DATE>:<BCX TYPE NUM>:<BCX REV>:<BCX DESC>:”

**Parameters:**

NAME	Friendly name for the BC4
MAC	Unique MAC address for the BC4
IP	Current IP address of the BC4
TCP PORT	The Port number used for TCP connections
UDP IN PORT	The Port used for listening to incoming UDP packets
UDP OUT PORT	The port used to send out UDP packets. Can be the same as the UDP IN PORT.
FIRMWARE V	Current Firmware revision
BUILD DATE	Current Firmware revision build date
BCX TYPE NUM	Specifies the specific type of BCX module which is currently installed. A “1” here would indicate a BCX-1
BCX REV	Indicates the hardware revision of the currently installed BCX module
BCX DESC	A friendly description of the currently installed BCX module

## BitWise Controls BC4 / BCX Command Protocol

---

**Example:** "bwc:discover:"

**Status:** "bws:discover:BITWISE00000000:00-50-C2-87-A0-03:  
192.168.1.6:5000:5200:5201:v1.5:Jun 09'2008:1:1.0:Serial, IR, GPIO, Relay:"

### Locate

**Purpose:**

Send a "locate" packet to a specific BC4 to turn on or off its Locate feature, which is a steady blink of its status LED. This is useful to physically locate a particular BC4 in a rack or installation.

**Format:**

"bwc:locate:<MODE>:"

**Returns:**

"bwr:locate:<MODE>:"

**Parameters:**

MODE	Indicates the current mode of the BC4's Locate feature, a "1" or "0" for On or Off respectively
------	---

**Example:**

"bwc:locate:1:"

**Reply:**

"bwr:locate:1:"

### Network Settings & Configuration

The following command packet types are used to apply and request the various network-related settings of a BC4

#### IPconfig

**Purpose:**

Used to Set or Get information BC4 IP settings

**Format:**

“bwc:<GET/SET>:ipconfig:<MAC>:(<NAME>:<DHCP>:<IP>:GATEWAY>:<SUBNET>:<DNS1>:<DNS2>:)”

**Returns:**

“bwr:<GET/SET>:ipconfig:<MAC>:<NAME>:<DHCP>:<IP>:<GATEWAY>:<SUBNET>:<DNS1>:<DNS2>:”

**Parameters:**

GET/SET	Indicates whether we are requesting or applying ipconfig settings
MAC	The MAC address of the BC4
DHCP	A flag to indicate whether or not to use DHCP
IP	IP Address of the BC4
GATEWAY	Gateway address
SUBNET	Subnet Mask
DNS1	First DNS address
DNS2	Second DNS address

**Example:**

“bwc:get:ipconfig:00-50-C2-87-A0-00:”

**Returns:**

“bwr:get:ipconfig: 00-50-C2-87-A0-00:TestBC4:0:192.168.1.100:192.168.1.1:255.255.255.0:100.100.100.1:100.100.100.2:”

**Example 2:**

“bwc:set:ipconfig:00-50-C2-87-A0-00:NewName:0:192.168.1.11:192.168.1.1:255.255.255.0:100.100.100.1:100.100.100.2:”

**Notes:**

If the DHCP flag is set to “1”, the rest of the fields can be left out, ie  
bwc:set:ipconfig:00-50-C2-87-A0-00:Name:1:

### PortConfig

**Purpose:**

Used to Get or Set the communication port numbers of the BC4

**Format:**

“bwc:<GET/SET>:portconfig:(<TCP>:<UDP IN>:<UDP OUT>:)”

**Returns:**

“bwr:<GET/SET>:portconfig:<TCP>:<UDP IN>:<UDP OUT>:”

**Parameters:**

GET/SET	Indicates whether we requesting or applying port config settings
TCP	TCP port number
UDP IN	UDP listen port
UDP OUT	UDP output port

**Example:**

“bwc:get:portconfig:”

**Returns:**

“bwr:get:portconfig:5000:5200:5201:”

**Example 2:**

“bwc:set:portconfig:5000:5200:5201:”

**Returns:**

“bwr:set:portconfig:5000:5200:5201:”

**Notes:**

The UDP IN and OUT ports can be the same number, but do not have to be.

### TCPTimeout

**Purpose:**

Used to set the amount of time (after the last incoming TCP packet) to wait before automatically closing a TCP connection. Since the BC4’s TCP server handles a single client at any one time, setting an automatic TCP timeout would be useful if there were more than one TCP host in an installation which wanted to access the BC4, or insuring that remote TCP connections do not get stuck in a “connected” state. Note, though, that automatically disconnecting from the host could mean that asynchronous status updates may be missed. For this reason, either choosing to use UDP broadcasting or using scheduled TCP “polling” events may be the desired methods to ensure that status updates are received in a multi-host system. The use of the “bwc:keepalive:” command can be used to keep the BC4 from automatically closing the TCP connection by resetting the tcptimeout timer.

## BitWise Controls BC4 / BCX Command Protocol

---

**Format:**

"bwc:<GET/SET>:tcptimeout:(<TIMEOUT>:)"

**Returns:**

"bwr:<GET/SET>:tcptimeout:<TIMEOUT>:"

**Status Update:**

"bws:tcptimedout:"

**Parameters:**

GET/SET	Indicates whether we requesting or applying port config settings
TIMEOUT	Time to wait ( in seconds, up to "255") after last activity before automatically disconnecting from TCP client. A "0" indicates that the BC4 will never automatically disconnect. Setting TCPTimeout to "0" can be dangerous in certain internet/port-forwarding applications, as the BC4 can hold a connection "open" even though the client has disconnected. It is advisable to use TCPTimeout in conjunction with KeepAlive and Heartbeat to let both the BC4 and the Host ensure that there is a viable connection.

**Example:**

"bwc:get:tcptimeout:"

**Returns:**

"bwr:get:tcptimeout:10:"

**Example 2:**

"bwc:set:tcptimeout:1:"

**Returns:**

"bwr:set:tcptimeout:1:"

**Status:**

"bws:tcptimedout:"

### Keepalive

**Purpose:**

Used to tell the BC4 to reset the tcptimeout timer.

**Format:**

"bwc:keepalive:"

**Returns:**

"bwr:keepalive:"

**Parameters: None**

**Example:**

“bwc:keepalive:”

**Returns:**

“bwr:keepalive:”

### TCPstatus

**Purpose:**

Used to determine the current state of the TCP connection.

**Format:**

“bwc:get:tcpstatus:”

**Returns:**

“bwr:get:tcpstatus:<STATUS>:”

**Parameters:**

STATUS	5 = waiting for incoming connection requests 6 = connected 7 = closed/ inactive
--------	---

**Example:**

“bwc:get:tcpstatus:”

**Returns:**

“bwr:get:tcpstatus:5:”

### TCPclose

**Purpose:**

Closes the current TCP session. It is a good idea to use this command when ending a session which is connected via the internet, to ensure the BC4 closes the socket immediately. Otherwise, the BC4 may appear unresponsive to connection requests while the socket recovers.

**Format:**

“bwc:tcpclose:”

**Returns:**

“bwr:tcpclose:”

**Parameters:**

None

### Heartbeat

**Purpose:**

Used to periodically alert the host that the BC4 is still present on the network.

## BitWise Controls BC4 / BCX Command Protocol

---

**Format:**

“bwc:<GET/SET>:heartbeat:<INTERVAL>:”

**Returns:**

“bwr:<GET/SET>:heartbeat:<INTERVAL>:<UPTIME>:”

**Status Update:**

“bws:heartbeat:<UPTIME>:”

**Parameters:**

GET/SET	Indicates whether we requesting or applying heartbeat settings
INTERVAL	Time to wait ( in seconds, up to “255”) between heartbeat notifications. A “0” indicates that the BC4 will never send a heartbeat status update
UPTIME	The number of seconds since the BC4 was reset

**Example:**

“bwc:get:heartbeat:”

**Returns:**

“bwr:get:heartbeat:10:120:”

**Example 2:**

“bwc:set:heartbeat:1:”

**Returns:**

“bwr:set:heartbeat:1:”

**Status:**

“bws:heartbeat:120:”

## UDPsendto

**Purpose:**

Sets or gets the IP address outgoing UDP packets will be sent to.

**Format:**

“bwc:<GET/SET>:udpsetto:<IP>:”

**Returns:**

“bwr: :<GET/SET>:udpsetto:<IP>:”

**Parameters:**

GET / SET	Indicates whether we requesting or applying the UDPsendto address
IP	The IP address that outgoing UDP packets will be sent to

**Example:**

“bwc:get:udpsetto:”

**Returns:**

“bwr:get:udpsetto:255.255.255.255:”

**Example 2:**

## BitWise Controls BC4 / BCX Command Protocol

---

`"bwc:set:udpsendto:192.168.1.121:"`

**Returns:**

`"bwr:set:udpsendto:192.168.1.121:"`

**Notes:**

Setting the UDPSendto address to "255.255.255.255" will result in all UDP packets being broadcast.

### UDPSend

**Purpose:**

Used to send a particular string to a UDP destination. The string will be sent to the UDPSendto address on the specified PORT (see UDPSendto command)

**Format:**

`"bwc:udpsend:<PORT>:<STRING>"`

**Returns:**

`"bwr:udpsend:<PORT>:<STRING>"`

**Parameters:**

PORT	The port to send the string on
STRING	The string of characters to be sent

**Example:**

`"bwc:udpsend:<PORT>:Hi, I'm a String of Characters"`

**Returns:**

`"bwr:udpsend:<PORT>:Hi, I'm a String of Characters"`

### RS (RemoteSend)

**Purpose:**

Used to send a command via UDP (on default UDPInPort 5200) to another BC4.

**Format:**

`"bwc:rs:<IP3>:<IP4>:<CMD>:"`

**Returns:**

`"bwr:rs:<IP3>:<IP4>:<CMD>:"`

**Parameters:**

IP3	The 3 <sup>rd</sup> part of the target IP address. For example, if the target was 192.168.1.10, the third part would be 1
IP4	The 4th part of the target IP address. For example, if the target was 192.168.1.10, the 4th part would be 10
CMD	Any valid bwc command, without the "bwc:" header

**Example:**

“bwc:rs:1:10:set:relay:1:1:”

**Returns:**

“bwr:rs:1:10:set:relay:1:1:”

## TCP Client

The following command packet types are used to allow the BC4 to act as a TCP client and interact with a TCP server device.

### TCPCopen

**Purpose:**

Used to open a client connection with a remote TCP server

**Format:**

“bwc:tcpcopen:<IP>:<PORT>:”

**Returns:**

“bwr: tcpcopen:<IP>:<PORT>:”

**Parameters:**

IP	The IP address of the remote TCP server
PORT	The TC P port to connect on

**Example:**

“bwc:tcpcopen:192.168.1.11:21:”

**Returns:**

“bwr:tcpcopen:192.168.1.11:21:”

### TCPCclose

**Purpose:**

Closes the current TCP client session

**Format:**

“bwc:tcpcclose:”

**Returns:**

“bwr:tcpcclose:”

**Parameters:**

None

### TCPCstatus

**Purpose:**

## BitWise Controls BC4 / BCX Command Protocol

---

Used to determine the current status of the TCP client connection.

**Format:**

"bwc:get:tcpstatus:"

**Returns:**

"bwr:get:tcpstatus:<STATUS>:"

**Parameters:**

STATUS	0 = waiting for incoming connection requests 1 = connected
--------	---

**Example:**

"bwc:get:tcpstatus:"

**Returns:**

"bwr:get:tcpstatus:5:"

### TCPCsendHex

**Purpose:**

Used to send the hex representation of a character or string of characters to the remote TCP server.

**Format:**

"bwc:tcpshendhex:<HEX>:"

**Returns:**

"bwr:tcpshendhex:<HEX>:"

**Parameters:**

HEX	The hex representation of the character, in pairs. For example, 41 = the character "A". 4141 = "AA"
-----	--

**Example:**

"bwc:tcpshendhex:41:"

**Returns:**

"bwr:tcpshendhex:41:"

### TCPCsendStr

**Purpose:**

Used to send an ASCII string to the remote TCP server.

**Format:**

"bwc:tcpshendstr:<STRING>:"

**Returns:**

"bwr:tcpshendstr:<STRING>:"

**Parameters:**

DECIMAL	The hex representation of the character. For example, "A" = the string "A"
---------	--

**Example:**

“bwc:tcpcsendstr:A:”

**Returns:**

“bwr:tcpcsendstr:A:”

## Temperature Sensor

### Temp

**Purpose:**

Gets a temperature reading from the BC4’s on-board temperature sensor

**Format:**

“bwc:get:temp:”

**Returns:**

“bwr:get:temp:<TEMP C>:<TEMP F>:<VAL>:”

**Status:**

“bws:get:temp:<TEMP C>:<TEMP F>:<VAL>:”

**Parameters:**

TEMP C	Degrees Celsius
TEMP F	Degrees Fahrenheit
VAL	The raw AD Value of the Temperature sensor

**Example:**

“bwc:get:temp:”

**Returns:**

“bwc:get:temp:40:105:281:”

**Status:**

“bws:get:temp:40:105:281:”

### TempConfig

**Purpose:**

Gets or sets the automatic status updating for the BC4’s on-board temperature sensor

**Format:**

“bwc:<GET / SET>:tempconfig:(<DELAY>:<OFFSET>:)”

**Returns:**

“bwr:<GET / SET>:tempconfig:<DELAY>:<OFFSET>:”

**Parameters:**

GET / SET	Indicates whether we are requesting or applying TempConfig settings
DELAY	The delay in 10ths of a second for automatic temperature updates. Setting the delay to “0” will disable automatic updating.

## BitWise Controls BC4 / BCX Command Protocol

OFFSET	The amount to offset the raw AD value to compensate for the BC4's internal temperature
--------	--

**Example:**

"bwc:get:tempconfig:"

**Returns:**

"bwr:get:tempconfig:100:0:"

**Example 2:**

"bwc:set:tempconfig:0:0:"

**Returns:**

"bwr:set:tempconfig:0:0:"

## Serial Ports

### SerialConfig

**Purpose:**

Gets or Sets the various settings for a given BCX module serial port.

**Format:**

"bwc:<GET / SET>:serialconfig:<SP>:<BAUD>:<TCP PORT>:"

**Returns:**

"bwc:<GET / SET>:serialconfig:<SP>:<BAUD>:<TCP PORT>:"

**Parameters:**

GET / SET	Indicates whether we are requesting or applying SerialConfig settings
SP	Specifies which Serial Port we are interested in
BAUD	Indicates the Baud rate being used 1 = 300, 2 = 1200, 3 = 2400, 4 = 4800, 5 = 9600, 6 = 19200, 7 = 38400, 8 = 57600, 9 = 115200
TCP PORT	Specifies what TCP port to use for the dedicated Serial port TCP socket

**Example:**

"bwc:get:serialconfig:1:"

**Returns:**

"bwr:get:serialconfig:1:9:5001:"

**Example 2:**

"bwc:set:serialconfig:1:5:5001:"

**Returns:**

"bwr:get:serialconfig:1:5:5001:"

### Send232Hex

**Purpose:**

Used to send the hex representation of a character or string of characters out the specified port.

**Format:**

"bwc:send232hex:<SP>:<HEX>:"

**Returns:**

"bwr:send232hex:<SP>:<HEX>:"

**Parameters:**

SP	Specifies the Serial Port to send the character to
DECIMAL	The hex representation of the character, in pairs, to be sent out the specified serial port. For example, 41 = "A". 4141 = "AA"

**Example:**

"bwc:send232hex:1:41:"

**Returns:**

"bwr:send232hex:1:41:"

### Send232str

**Purpose:**

Used to send an ASCII string out the specified port.

**Format:**

"bwc:send232str:<SP>:<STRING>"

**Returns:**

"bwr:send232str:<SP>:<STRING>"

**Parameters:**

SP	Specifies the Serial Port to send the String to
STRING	The ASCII string to be sent out the specified Serial Port. For example, "Hello World"

**Example:**

"bwc:send232str:1:Hello World"

**Returns:**

"bwr:send232str:1:Hello World"

### Send232termstr

**Purpose:**

## BitWise Controls BC4 / BCX Command Protocol

---

Used to send an ASCII string followed by one or more hex terminators out the specified port.

**Format:**

“bwc:send232termstr:<SP>:<TERM>:<STRING>”

**Returns:**

“bwr:send232termstr:<SP>:<TERM>:<STRING>”

**Parameters:**

SP	Specifies the Serial Port to send the String to
TERM	One or more Hex values which will be sent AFTER the string value. For example, “0D” would send a Carriage Return. “0D0A” would send Carriage Return, Line Feed.
STRING	The ASCII string to be sent out the specified Serial Port. For example, “Hello World”

**Example:**

“bwc:send232termstr:1:0D:Hello World”

**Returns:**

“bwr:send232termstr:1:0D:Hello World”

### Baud

**Purpose:**

Used to get or set the baud rate for a given BCX module serial port

**Format:**

“bwc:<GET / SET>:baud:<SP>:(<BAUD>:)”

**Returns:**

“bwr:<GET / SET>:baud:<SP>:<BAUD>:”

SP	Specifies the Serial Port to send the String to
BAUD	Indicates the Baud rate being used 1 = 300, 2= 1200, 3 = 2400, 4 = 4800, 5 = 9600, 6 = 19200, 7 = 38400, 8 = 57600, 9 = 115200

**Example:**

“bwc:get:baud:1:”

**Returns:**

“bwr:get:baud:1:9:”

**Example 2:**

“bwc:set:baud:1:8:”

**Returns:**

“bwr:set:baud:1:8:”

### TCPSclose

**Purpose:**

Closes the serial port's TCP client session

**Format:**

"bwc:tcpsclose:<SP>:"

**Returns:**

"bwr:tcpsclose:<SP>:"

**Parameters:**

SP	Specifies the Serial Port number to close
----	---

### TCPSstatus

**Purpose:**

Used to determine the current status of the TCP Serial Port connection.

**Format:**

"bwc:get:tcpsstatus:<SP>:"

**Returns:**

"bwr:get:tcpsstatus:<SP>:<STATUS>:"

**Parameters:**

SP	Specifies the Serial Port number to close
STATUS	0 = waiting for incoming connection requests 1 = connected

**Example:**

"bwc:get:tcpsstatus:1:"

**Returns:**

"bwr:get:tcpsstatus:1:0:"

## GPIO

### IOconfig

**Purpose:**

Gets or sets the various settings for a given BCX module GPIO port

**Format:**

## BitWise Controls BC4 / BCX Command Protocol

---

“bwc:<GET / SET>:ioconfig:(<GPIO>:<MODE>:<DELAY>:<THRESH>:)”

**Returns:**

“bwr:<GET / SET>:ioconfig:<GPIO>:<MODE>:<DELAY>:<THRESH>:”

GET / SET	Indicates whether we are requesting or applying GPIO configuration settings
GPIO	Specifies the GPIO we are interested in
MODE	Specifies the operation mode to use. 0 = Disabled, 1 = Digital Input, 2 = Analog Input, 3 = Digital Output, 4 = Relay, 5 = Counter, 6 = Frequency
DELAY	Specifies the delay (in 10ths of a second) between GPIO status updates (assuming the status has changed during the delay period). A delay of “0” will disable automatic status updates.
THRESH	Specifies the Digital Threshold which determines the difference between On and Off states. Valid range is “0” to “102” where “0” = 0V, “102” = Max AD input voltage. Check the BCX module datasheet for the maximum accepted voltage.

**Example:**

“bwc:get:ioconfig:1:”

**Returns:**

“bwr:get:ioconfig:1:2:1:1:”

**Example 2:**

“bwc:set:ioconfig:1:1:1:10:”

**Returns:**

“bwr:set:ioconfig:1:1:1:10:”

**Notes:**

Not all MODES are usable with all GPIOs. For specific information, please refer to the datasheet for the BCX module in use.

### Input

**Purpose:**

Gets the current state of a given BCX module GPIO which has been configured as a Digital Input (Mode 1)

**Format:**

“bwc:get:input:<GPIO>:”

**Returns:**

“bwr:get:input:<GPIO>:<STATE>:”

**Status Update:**

“bws:get:input:<GPIO>:<STATE>:”

**Parameters:**

GPIO	The GPIO number of a BCX module GPIO port which has been configured as a
------	--

## BitWise Controls BC4 / BCX Command Protocol

	Digital Input
STATE	The current state of the GPIO. "0" = Off, the voltage present is less than the Threshold setting for the GPIO "1" = On, the voltage present is greater than or equal to the Threshold setting for the GPIO

**Example:**

"bwc:get:input:1:"

**Returns:**

"bwr:get:input:1:1:"

**Status:**

"bws:get:input:1:1:"

### AD

**Purpose:**

Gets the current values associated with a BCX module GPIO port which has been configured as an Analog Input (Mode 2)

**Format:**

"bwc:get:ad:<GPIO>:"

**Returns:**

"bwr:get:ad:<GPIO>:<CUR>:<MIN>:<MAX>:"

**Status Update:**

"bws:get:ad:<GPIO>:<CUR>:<MIN>:<MAX>:"

**Parameters:**

GPIO	The GPIO number of a BCX module GPIO port which has been configured as an Analog Input
CUR	The current value present at the GPIO
MIN	The minimum value measured since the last update
MAX	The maximum value measured since the last update

**Example:**

"bwc:get:ad:1:"

**Returns:**

"bwr:get:ad:1:821:803:856:"

**Status:**

"bws:get:ad:1:821:803:856:"

**Notes:**

Check the datasheet associated with the specific BCX module used to see the AD resolution and value range for your GPIO.

### Output

**Purpose:**

Gets or sets the state of a BCX module GPIO which has been configured as a Digital Output (Mode 3)

**Format:**

"bwc:<GET / SET>:output:<GPIO>:<STATE>:"

**Returns:**

"bwr:<GET / SET>:output:<GPIO>:<STATE>:"

**Status Update:**

"bws:get:output:<GPIO>:<STATE>:"

**Parameters:**

GET / SET	Indicates whether we are requesting or applying the state of a BCX module GPIO port which has been configured as a Digital Output
GPIO	The GPIO number of a BCX module GPIO port which has been configured as an Digital Output
STATE	The state of the output. "0" = Off, "1" = On

**Example:**

"bwc:get:output:1:"

**Returns:**

"bwr:get:output:1:1:"

**Status:**

"bws:get:output:1:1:"

**Notes:**

See the "Pulse" command for further output functionality

### Count

**Purpose:**

Gets or sets the Counter value of a BCX module's Counter, if present.

**Format:**

"bwc:<GET / SET>:count:<VALUE>:"

**Returns:**

"bwr:<GET / SET>:count:<VALUE>:"

**Status Update:**

"bws:get:count:<VALUE>:"

**Parameters:**

GET / SET	Indicates whether we are requesting or applying the value of a BCX module's Counter input
VALUE	The Value of the Counter input. Can be set to "0" to reset the counter

**Example:**

"bwc:get:count:"

**Returns:**

"bwr:get:count:126:"

**Example 2:**

"bwc:set:count:0:"

**Returns:**

"bwr:set:count:0:"

**Status:**

"bws:get:count:0:"

**Notes:**

Refer to the User's Guide for your BCX module for more specific information regarding the Counter functionality.

### Frequency

**Purpose:**

Gets or sets the value of a BCX module's Frequency Input, if present.

**Format:**

"bwc:get:freq:(<VALUE>:)"

**Returns:**

"bwr:get:freq:<VALUE>:"

**Status Update:**

"bws:get:freq:<VALUE>:"

**Parameters:**

VALUE	The Value of the Counter input. Can be set to "0" to reset the counter
-------	--

**Example:**

"bwc:get:freq:"

**Returns:**

"bwr:get:freq:0:"

**Status:**

"bws:get:freq:132:"

**Notes:**

Refer to the User's Guide for your BCX module for more specific information regarding the Frequency Input functionality.

### Relays

#### Relay

**Purpose:**

Gets or sets the state of a given BCX module Relay

**Format:**

"bwc:<GET / SET>:relay:<RELAY>:<STATE>:"

**Returns:**

"bwr :<GET / SET>:relay:<RELAY>:<STATE>:"

**Status Update:**

"bws:get:relay:<RELAY>:<STATE>:"

**Parameters:**

GET / SET	Indicates whether we are requesting or applying the state of a BCX module Relay
RELAY	Specifies the Relay we are interested in
STATE	The state of the Relay. "0" = Off, "1" = On.

**Example:**

"bwc:get:relay:1:"

**Returns:**

"bwr:get:relay:1:0:"

**Example 2:**

"bwc:set:relay:1:1:"

**Returns:**

"bwc:set:relay:1:1:"

**Status:**

"bws:get:relay:1:1:"

#### Pulse

**Purpose:**

Pulses a given BCX module Relay or Digital Output for a given length of time.

**Format:**

"bwc:set:pulse:<TYPE>:<NUM>:<PULSE>:"

**Returns:**

"bwr:set:pulse:<TYPE>:<NUM>:<PULSE>:"

**Parameters:**

TYPE	Indicates what type of output we are pulsing. "1" = A BCX module GPIO which has been configured as a Digital Output "2" = A BCX module Relay
------	--

## BitWise Controls BC4 / BCX Command Protocol

NUM	The GPIO or Relay number to be pulsed
PULSE	The pulse-width, in milliseconds

**Example:**

`"bwc:set:pulse:2:1:200:"`

**Returns:**

`"bwr:set:pulse:2:1:200:"`

## IR (Infrared)

The IR command has many different Modes which dictate exactly what parameters will make up a valid packet. Although they each have the command Type of "IR", the different modes of the IR command Type will be treated as separate commands for the purposes of this document. Note that a Status update will be sent every time the IR engine changes state, or is asked to change state while busy.

### SendIR

**Purpose:**

Sends a Library IR code pulse from a given IR port

**Format:**

`"bwc:ir:0:1:<PORT>:<DEVTYPE>:<DEVCODE>:<KEYNUM>:"`

**Returns:**

`"bwc:ir:<IRSTATE>:1:"`

**Status Update:**

`"bws:ir: :<IRSTATE>:1:"`

**Parameters:**

IRSTATE	The state of the IR engine, indicating the result of the requested state change. "0" = Success, "1" = Invalid Device Code, "2" = Invalid Device Type, "3" = Bad Key or No IR, "6" = Error or Timeout, "7" = Data Format Error
PORT	The BCX module IR port to use
DEVTYPE	The Device Type number of the IR code to send
DEVCODE	The Device Codeset number of the IR code to send
KEYNUM	The IR Key number to send

**Example:**

`"bwc:ir:0:1:1:6:490:13:"`

**Returns:**

"bwr:0:1:"

**Status:**

"bws:0:1:"

**Notes:**

The valid Device Type, Device Codeset, and IR Key numbers are documented in the IR Library documents available at [www.bitwisecontrols.com](http://www.bitwisecontrols.com)

### SendIR (continuous)

**Purpose:**

Sends a Library IR code continuously until canceled or timed out from a given IR port

**Format:**

"bwc:ir:0:2:<PORT>:<DEVTYPE>:<DEVCODE>:<KEYNUM>:"

**Returns:**

"bwc:ir:<IRSTATE>:2:"

**Status Update:**

"bws:ir: :<IRSTATE>:2:"

**Parameters:**

IRSTATE	The state of the IR engine, indicating the result of the requested state change. "0" = Success, "1" = Invalid Device Code, "2" = Invalid Device Type, "3" = Bad Key or No IR, "6" = Error or Timeout, "7" = Data Format Error
PORT	The BCX module IR port to use
DEVTYPE	The Device Type number of the IR code to send
DEVCODE	The Device Codeset number of the IR code to send
KEYNUM	The IR Key number to send

**Example:**

"bwc:ir:0:2:1:6:490:13:"

**Returns:**

"bwr:0:2:"

**Status:**

"bws:0:2:"

**Notes:**

The valid Device Type, Device Codeset, and IR Key numbers are documented in the IR Library documents available at [www.bitwisecontrols.com](http://www.bitwisecontrols.com)



## BitWise Controls BC4 / BCX Command Protocol

PORT	The BCX module IR port to use
------	-------------------------------

**Example:**

```
"bwc:ir:0:4:1:4E01014A00E20A0062882701E00518005404E5011A034000620319002A0
35000C501D700E200DB007F013E003801853C177663778889999589589554552076677
7777777777:"
```

**Returns:**

```
"bwr:ir:0:4:"
```

**Status:**

```
"bws:ir:0:4:"
```

### Reset

**Purpose:**

Resets the BCX module's IR engine

**Format:**

```
"bwc:ir:0:5:"
```

**Returns:**

```
"bwr:ir:0:5:"
```

**Status:**

```
"bws:ir:0:5:"
```

### Cancel

**Purpose:**

Cancels a continuous IR transmission

**Format:**

```
"bwc:ir:0:8:"
```

**Returns:**

```
"bwr:ir:<IRSTATE>:8:"
```

**Status Update:**

```
"bws:ir:<IRSTATE>:8:"
```

**Parameters:**

IRSTATE	The state of the IR engine, indicating the result of the requested state change. "0" = Success "7" = Data Format Error, or there was no continuous IR to cancel
---------	---

**Example:**

```
"bwc:ir:0:8:"
```

**Returns:**

```
"bwr:ir:0:8:"
```

**Status:**

## BitWise Controls BC4 / BCX Command Protocol

---

“bws:ir:0:8:”

### GetKeys

**Purpose:**

Gets the available valid key codes for a given Device Type and Device Codeset

**Format:**

“bwc:ir:0:7:<DEVTYPE>:<DEVCODE>:”

**Returns:**

“bwr:ir:<IRSTATE>:7:<DEVTYPE>:<DEVCODE>:<KEYS>:”

**Status Update:**

“bws:ir:<IRSTATE>:7:”

**Parameters:**

IRSTATE	The state of the IR engine, indicating the result of the requested state change. “0” = Success, “1” = Invalid Device Code, “2” = Invalid Device Type, “6” = Error or Timeout, “7” = Data Format Error
DEVTYPE	The Device Type number of the IR code to send
DEVCODE	The Device Codeset number of the IR code to send
KEYS	The Valid Keys

**Example:**

“bwc:ir:0:7:6:”

**Returns:**

“bwr:ir:0:7:6:490:2F12090A0B0C0D0E0F10110405010203181A1B1C191D2221262728292A1E1F30312E1523332425322F2D3E432B1417:”

**Status:**

“bws:0:7”

**Notes:**

The KEYS string should be read as a string of HEX bytes, the first byte being the length of the string, including itself. In the above example, the first portion of the KEYS string is “2F12090A.....”. This should be interpreted as “0x2F,0x12,0x09,0x0A.....” The first byte, 0x2F tells us that the string should include 47 bytes, or 46 valid keys plus the length byte. The first three valid keys in this string would be 18 (0x12), 9 (0x09), and 10 (0x0A).

### Learn

**Purpose:**

Learn an IR code via the BCX module’s on-board IR learner.

**Format:**

“bwc:ir:0:6:”

**Returns:**

“bwc:ir:<IRSTATE>:6:<IR>:”

## BitWise Controls BC4 / BCX Command Protocol

---

**Status:**

"bws:ir:<IRSTATE>:6:"

**Parameters:**

IRSTATE	The state of the IR engine, indicating the result of the requested state change. "0" = Success, "6" = Error or Timeout, "7" = Data Format Error
IR	The Learned IR data, if learn was successful

**Example:**

"bwc:ir:0:6:"

**Returns:**

"bwc:ir:0:6:3401013000E10700E18886053701C0029401C601C201B901C200DA00E101B  
B00E100DCA016655266666666636636666366664540:"

**Status:**

"bws:ir:0:6:"

## Variables

### State

**Purpose:**

Gets or sets the state of a given user variable. Note that variables 0 – 29 correspond with BC4 variables 57-86

**Format:**

"bwc:<GET / SET>:state:<VARIABLE>:(<STATE>:)"

**Returns:**

"bwr :<GET / SET>:relay:<VARIABLE>:<STATE>:"

**Parameters:**

GET / SET	Indicates whether we are requesting or applying the state of a BCX module Relay
VARIABLE	Specifies the variable we are interested in
STATE	The state of the variable, 0-255.

**Example:**

"bwc:get:state:57:"

**Returns:**

"bwr:get:state:57:0:"

**Example 2:**

"bwc:set:state:57:1:"

**Returns:**

"bwr:set:state:57:1:"

## Macros

### Execute Macro by Number

**Purpose:**

Execute a BC4 macro which has been stored in the BC4 with BC4 Project Editor software

**Format:**

"bwc:macro:<MACRO>:"

**Returns:**

"bwr:macro:<MACRO>:"

**Parameters:**

MACRO	The macro number to be executed
-------	---------------------------------

**Example:**

"bwc:macro:3:"

**Returns:**

"bwr:macro:3:"

### Execute Macro by Name

**Purpose:**

Execute a BC4 macro which has been stored in the BC4 with BC4 Project Editor software. Note that this method is new and requires BC4 firmware 1.068 or higher, and macros generated by BC4 Project Editor version 1.0.5 or higher.

**Format:**

"bwc:macrobyname:<MACRONAME>:"

**Returns:**

"bwr:macrobyname:<MACRONAME>:"

**Parameters:**

MACRONAME	The macro name to be executed
-----------	-------------------------------

**Example:**

"bwc:macrobyname:Watch DVD:"

**Returns:**

"bwr:macrobyname:Watch DVD:"

### Get Macro Name

**Purpose:**

Retrieve the name of a stored macro.

Note that this method is new and requires BC4 firmware 1.068 or higher, and macros generated by BC4 Project Editor version 1.0.5 or higher. Applications may use this command to loop through numbers 1-500, storing any returned macro names. BC4 will NAK if there is no macro stored at the given location.

**Format:**

"bwc:get:macro:<MACRONUMBER>:2:"

**Returns:**

"bwr:macro:<MACRONUMBER>:2:<MACRONAME>"

**Parameters:**

MACRONUMBER	The macro location to retrieve. Valid numbers are 1-500.
MACRONAME	The macro name stored at MACRONUMBER

**Example:**

"bwc:get:macro:6:2:"

**Returns:**

"bwr:macro:6:2:Watch DVD:"

## Misc

### Reset

**Purpose:**

Reboots the BC4

**Format:**

"bwc:reset:"

**Returns:**

"bwr:reset:"

### Heartbeat

**Purpose:**

## BitWise Controls BC4 / BCX Command Protocol

---

Defines a “heartbeat” signal to signify that a BC4 is still present and operational.

**Format:**

“bwc:<GET / SET>:heartbeat:<INTERVAL>:(<LAST>:)”

**Returns:**

“bwc:<GET / SET>:heartbeat:<INTERVAL>:<LAST>:”

**Status:**

“bws:get:heartbeat:<LAST>:”

**Parameters:**

GET / SET	Indicates whether we are requesting or applying heartbeat properties
INTERVAL	The amount of time (in seconds) between heartbeat status updates. Set to “0” to disable status updates.
LAST	The amount of time (in seconds) since the BC4 or INTERVAL value were reset.

**Example:**

“bwc:get:heartbeat:”

**Returns:**

“bwc:get:heartbeat:10:120:”

**Example 2:**

“bwc:set:heartbeat:10:”

**Returns:**

“bwc:set:heartbeat:10:110:”

**Status:**

“bws:heartbeat:140:”

### DelayedCmd

**Purpose:**

Allows a command to be executed after a specified delay period

**Format:**

“bwc:<GET / SET>:delayedcmd:<TIMER>:(<INTERVAL>:<CMD>:)”

**Returns:**

“bwc:<GET / SET>:delayedcmd:<TIMER>:<INTERVAL>:(<CMD>:)”

**Parameters:**

GET / SET	Indicates whether we are requesting or applying a delayed command
-----------	---

## BitWise Controls BC4 / BCX Command Protocol

TIMER	The delay timer to use. Can be "1" or "2" .
INTERVAL	The amount of time (in tenths of a second, "0" – "65535") to wait before executing CMD. Set to "0" to cancel a previously set delayed command. When retrieved via a "get", will be the amount of time remaining until CMD is executed.
CMD	The valid command to execute

**Example:**

"bwc:get:delayedcmd:1:"

**Returns:**

"bwr:get:delayedcmd:1:120:bwc:set:relay:1:1:"

**Example 2:**

"bwc:set:delayedcmd:1:10:bwc:set:relay:1:0:"

**Returns:**

"bwr:set:delayedcmd:1:10:bwc:set:relay:1:0:"

## Resources

There are many support resources available in our support section at [www.bitwisecontrols.com](http://www.bitwisecontrols.com)  
Be sure to check in there for product news, software downloads and updates, and much more.

## Contact Information

We want to hear any questions and comments you may have about our company and products.

Please feel free to contact us.

[www.bitwisecontrols.com](http://www.bitwisecontrols.com)

**BitWise Controls, LLC**

188 Inverness Drive West  
Suite 310

Englewood, CO 80112

Phone: 866.932.1BWC (2292)

Fax: 866.932.2292