iVu BCR Communications

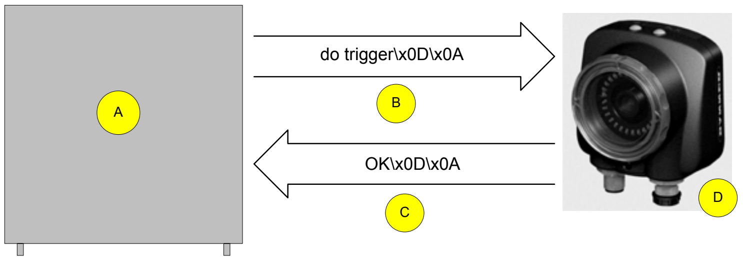
Instruction Manual

Original Instructions
B_3074177 Rev. C
4 March 2014

Contents

1 Command Channel

The iVu BCR command channel is a bi-directional communication protocol that currently supports ASCII via the RS-232 serial interface, and enables other devices to remotely control the iVu sensor and access sensor results.



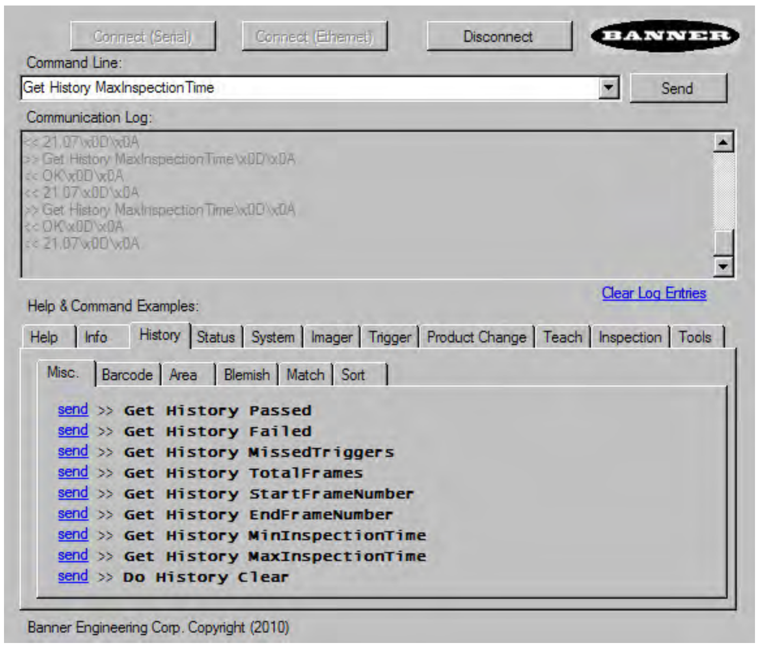
A	Control Device, which can be a PLC, PC program, or a terminal	C	Response Frame
B	Request Frame	D	iVu BCR Sensor

The following are some of the functionality available via the command channel:

- Get sensor information (such as version and sensor name)
- Control "discrete" I/O (such as trigger and teach)
- Get sensor results (such as sensor status)
- Set and get BCR compare strings

Command Channel Sample Application

The iVu BCR installation CD has a Command Channel sample application that provides an easy interface to execute commands. In a production environment, you will need to create your own application for bi-directional communication with the sensor.



Command Line: Get History MaxInspectionTime

```
<< 21.07\x0D\x0A
>> Get History MaxInspectionTime\x0D\x0A
<< OK\x0D\x0A
<< 21.07\x0D\x0A
>> Get History MaxInspectionTime\x0D\x0A
<< OK\x0D\x0A
<< 21.07\x0D\x0A
```

```
send >> Get History Passed
send >> Get History Failed
send >> Get History MissedTriggers
send >> Get History TotalFrames
send >> Get History StartFrameNumber
send >> Get History EndFrameNumber
send >> Get History MinInspectionTime
send >> Get History MaxInspectionTime
send >> Do History Clear
```

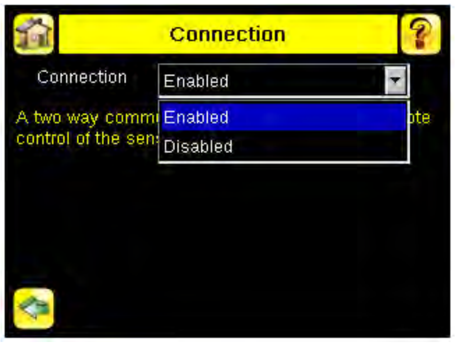
Banner Engineering Corp. Copyright (2010)

2 Setting Up Serial Communications

1. Electrically connect the control device and the iVu sensor. On the iVu, the pins/wire colors used for serial communications via RS-232 are shown in the table below.

iVu RS-232 Connections		
Pin #	Wire Color	Description
10	Light-Blue	TX
11	Black	Signal Ground
12	Violet	RX

2. Enable the command channel. Go to Main Menu > System > Communications > Command Channel > Connection .

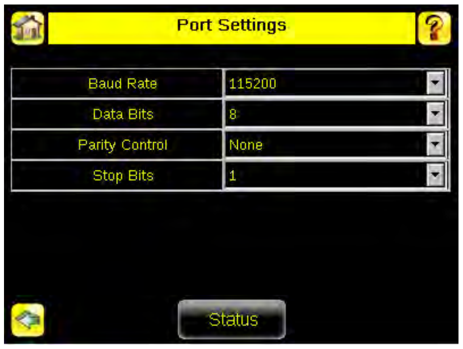


Connection

Connection: Enabled

A two way comm te
control of the sen

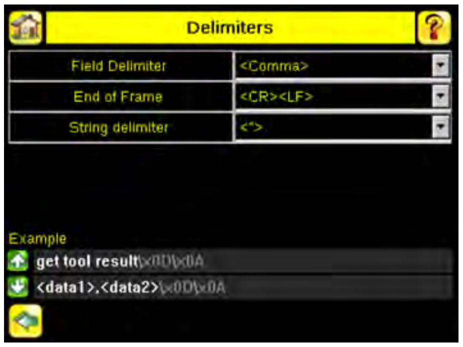
3. Configure port settings (baud rate, data bits, parity, and stop bits) on the iVu to match the settings on the control device. Go to Main Menu > System > Communications > Serial I/O .



Port Settings

Baud Rate	115200
Data Bits	8
Parity Control	None
Stop Bits	1

4. Configure end-of-frame delimiters. Go to Main Menu > System > Communications > Command Channel > Delimiters .



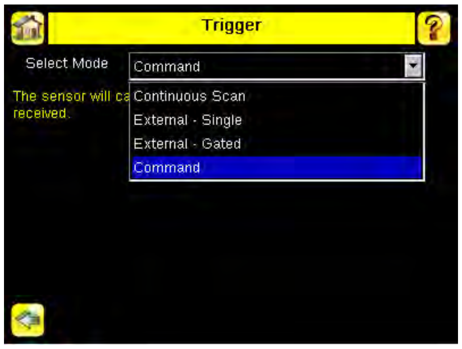
Delimiters

Field Delimiter	<Comma>
End of Frame	<CR><LF>
String delimiter	<">

Example
get tool result\x0D\x0A
<data1>,<data2>\x0D\x0A

Valid end-of-frame delimiters are: <comma>, <colon>, <semicolon>, <CR>, <CR><LF>, <LF><CR>, or <ETX>.

5. Optionally, if you want to trigger the iVu from the control device, set the trigger mode to Command (go to Main Menu > Imager > Trigger and select Command from the drop-down).



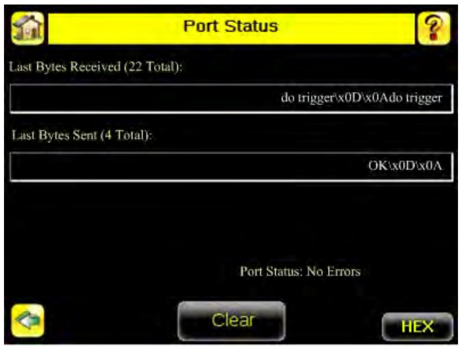
6. Verify that the iVu receives and transmits data correctly.

3 Testing and Troubleshooting iVu Command Channel Communications

3.1 Using the Port Status Screen for Testing RS-232 Communications

The Port Status screen can be used to ensure data is entering and exiting the sensor. This can be useful for debugging issues such as improper wiring, mismatched baud rates, or other serial I/O issues. To access the Port Status screen, go to Main Menu > System > Communications > Serial I/O and click on the Status button.

- The upper field shows the bytes received (request frame) on the iVu from the control device.
- The lower field shows the bytes sent (response frame) from the iVu to the control device.



Last Bytes Received (22 Total):

do trigger\x0D\x0Ado trigger

Last Bytes Sent (4 Total):

OK\x0D\x0A

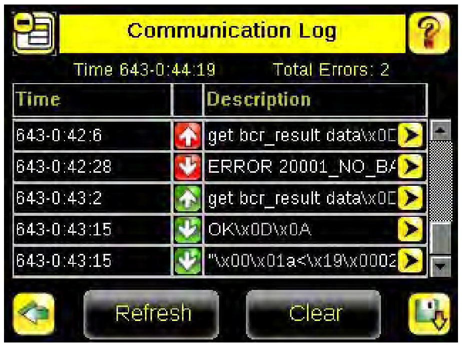
Port Status: No Errors

3.1.1 Port Errors

The Port Errors screen can help to debug communications channel issues: Parity, Break, and Framing indicate mismatched port settings or, in the case of Break, incorrect cabling.

3.2 Understanding the Communication Log

The Communication Log can be used to ensure commands are properly formed (syntax is correct), and provides a history of commands issued along with responses to these commands. To access the Communication log, go to Main Menu > Logs > Communication Log .



Time 643-0:44:19 Total Errors: 2

Time		Description
643-0:42:6	↑	get bcr_result data\x0D
643-0:42:28	↓	ERROR 20001_NO_BA
643-0:43:2	↑	get bcr_result data\x0D
643-0:43:15	↓	OK\x0D\x0A
643-0:43:15	↓	"\x00\x01a<\x19\x0002

Some notes about the logs:

- To see an expanded view of each entry, click on the small right-triangle control on each entry
- To save the log, click the save icon. The saved communication log can be loaded into the emulator for troubleshooting offline

The table below describes the icons used in the Communication Log, the up-arrow indicates an incoming request to the iVu from the control device; the down-arrow indicates an outgoing response from the iVu to the control device.

Icon	Description
●	Port opened.
●	Port closed.
⬆ ⬇	Indicates that the command has been processed without errors.
⬆ ⬇	Indicates that the incoming entry is stalled (no new bytes), or end-of-frame delimiter was not received .
⬆ ⬇	If the response frame contains an error or is dropped, the log entry icons for the request and the response frames will be colored red, and the displayed error count will increment by one.
⌛	If the command takes a long time to process, the last long entry will change to an hourglass (for example, during trigger of long inspections).

3.3 Using the iVu Command Channel Sample Application or a Terminal Program for Testing

The easiest way to test that the iVu command channel is correctly receiving and transmitting data is to use either the iVu Command Channel Sample App (available on the installation CD) or to use a terminal program running on a PC:

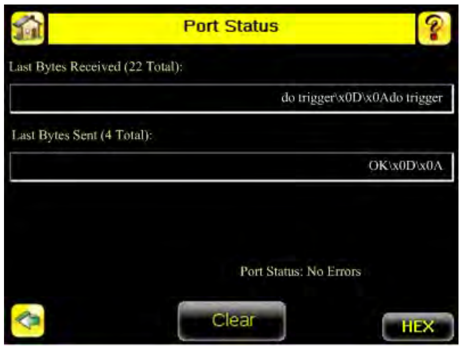
If using a terminal program, in the terminal program's configuration:

- Set new-line transmit to <CR><LF> (and set the end-of-frame delimiters on the iVu to match).
- Enable local echo.
- Set the Serial port set up so that the PC port number's baud rate, data, parity, and stop bits match those setup on the iVu.

3.4 Verifying Basic Receive Functionality

To verify the iVu can receive request frames from the requesting device:

1. On the iVu Sensor, go to the Main Menu > System > Communications > Serial I/O > Port Status screen.



Last Bytes Received (22 Total):
do trigger\x0D\x0Ado trigger
Last Bytes Sent (4 Total):
OK\x0D\x0A
Port Status: No Errors

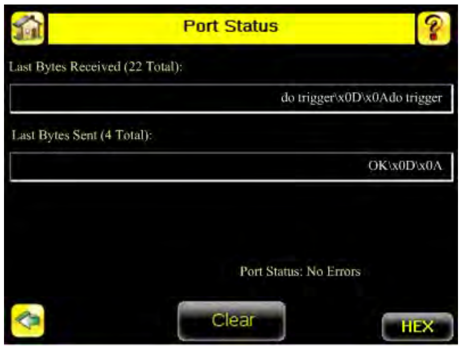
2. On the requesting device, transmit one or more bytes to the iVu sensor.
   - If the data byte values appear correct and the number sent by the requesting device matches the number received by the iVu sensor, then the transmit/receive functionality is working properly.
   - If the connection is incorrect (electrically) or if the baud rate is mismatched, no bytes will appear in the upper field on the Port Status screen.
   - If the connection is correct (electrically), bytes will appear in the upper field of the Port Status screen in the order they were received.
   - If the Port Status: Errors at the bottom of the Port Status screen highlights red, then the connection is correct electrically but there is likely a settings mismatch between the iVu sensor and the requesting device. Verify the settings on both devices.
   - If the bytes appear with no errors but appear incorrect or there are too many or too few, then the port settings (for example, baud rate) are likely mismatched in a way that does not generate serial hardware errors. Verify the settings on both devices match exactly.

3.5 Verifying Basic Transmit Functionality

The iVu command channel will only send response frames to the requesting device if it receives a valid end-of-frame delimiter from the requesting device. To verify transmit functionality:

1. Transmit an end-of-frame delimiter sequence from the requesting device to the iVu sensor. For example, in a terminal program, simply hit Enter.

   If a valid end-of-frame delimiter is received, the iVu sensor will immediately transmit a short error message back to the requesting device (for example, ERROR 10000_COMMAND_MISSING).
2. Verify that the number of bytes sent by the requesting device are the same as the number shown in the lower field of the Port Status screen on the iVu sensor. Go to the Main Menu > System > Communications > Serial I/O > Port Status screen.



Last Bytes Received (22 Total):

do trigger\x0D\x0Ado trigger

Last Bytes Sent (4 Total):

OK\x0D\x0A

Port Status: No Errors

3. If the byte count does not match, re-verify that the settings on both devices match exactly. If no bytes are received, re-check the wiring.

If the correct response frame is received, then basic electrical and port settings are correct.

4 Command Channel Commands

All iVu command channel request command frames use the following syntax:

```
>> command group item value<EOF>
```

Notes
: <EOF> is the end-of-frame delimiter. See below for a description.
All commands are in ASCII and are case-insensitive

command
: An action to be performed on a particular iVu group; for example, get, set, do, login, or logout.

group
: Identifies the iVu group that the command should act upon; for example, info, system, trigger, or bcr_input.

item
: Further qualifies the action by specifying an item within the identified group; for example, comparedata or status.

value
: For set commands, this identifies the data that must be set for the specified group item.

Note: Item is not used with get commands.

<EOF>
: Identifies the end-of-frame for the command so that the iVu knows to begin processing. The iVu will return a response that includes the end-of-frame delimiter. The options for the <EOF> are set in the iVu Serial I/O menu, and are as follows:

- <comma>
- <colon>
- <semicolon>
- <CR>
- <CR><LF>
- <LF><CR>
- <ETX>

NOTE: When data is displayed on iVu screens such as the Port Status screen, printable delimiters are displayed as expected. Non-printable characters, such as <CR> are displayed in hex notation (\x0D).

4.1 Command Flow

The command flow should be such that a new command request should not be issued until the iVu command channel acknowledges the previous command request.

For example, the following is a series of command requests and responses. The first request sets the trigger mode to command and, once the sensor responds with an "OK," the next command request is issued to do (or execute) the trigger.

```
>>  set trigger mode command\x0D\x0A
<<  OK\x0D\x0A
>>  do trigger\x0D\x0A
<<  OK\x0D\x0A
```

4.2 String Delimiters and Escaping

By default setting, all strings used in commands are enclosed in quotation marks (""). All text in quotes is part of the command. Quotes (") or back-slashes (\) that are part of the string must be escapted with a back-slash. For example:

"abc\"def\"ghi\\jkl"

Set the String Delimiter parameters to 'None' if strings should not be enclosed in quotation marks.

4.3 Command Channel Command Synopsis

There are a number of general types of commands to do, set, and get sensor data.

4.3.1 Do Commands

Do commands are actions (methods) to perform on the sensor, such as trigger, reboot, and the like.

Command	Group	Description
Reboot	System	Reboots the sensor. Pre-empts other commands except Save.
Save	System	Saves inspection and configuration parameters. Blocks until finished. Should be used sparingly.
ClearSystemError	Status	Clears the system error LED and sets the internal flag to false.
Immediate	Trigger	Initiates a single trigger. The sensor does not transmit a response unitl the sensor has completed the action.
Gated	Trigger	Initiates gated triggering. The sensor does not transmit a response unitl the sensor has completed the action.
AbortGated	Trigger	Aborts gated triggering. The sensor does not transmit a response unitl the sensor has completed the action, then it will respond with two responses.
NextTrigger	Teach	Sets the sensor to teach on the next trigger
Clear	History	Clears all history fields (for example pass, fail, etc.).

4.3.2 Get Commands

Get commands are used to retrieve information from the sensor (for example, get the status of the sensor).

Command	Group	Description
CompanyName	Info	The company name as a string.
ModelNumber	Info	The sensor model number as a string.
FirmwareVersion	Info	The sensor firmware version as a string.
SerialNumber	Info	The sensor serial number as a string.
Name	Info	The sensor name as a string.
BootNumber	Info	The number of sensor bootups
UpTimer	Info	The elapsed time the sensor has been running in the format hh:mm:ss:msec.
HourCount	Info	The number of hours the sensor has been running.
RemoteConnected	Info	The remote display connected status as a boolean value (true or false)
RemoteModelNumber	Info	The model number of the remote display as a string.
RemoteSerialNumber	Info	The serial number of the remote display as a string.
Ready	Status	Flag indicating whether the system is ready to trigger (true) or busy (false)
SystemError	Status	Flag indicating whether a system error is active (true) or cleared (false)

Command	Group	Description
Mode	Trigger	Sets trigger mode to one of the valid trigger modes for the sensor.
Passed	History	The number of passed inspections.
Failed	History	The number of failed inspections.
MissedTriggers	History	The number of missed triggers.
StartFrameNumber	History	The starting frame number.
EndFrameNumber	History	The ending frame number.
MinInspectionTime	History	The minimum elapsed time (msec) of the inspection.
MaxInspectionTime	History	The maximum elapsed time (msec) of the inspection.
MinBarcodeCount	History	The minimum number of barcodes read.
MaxBarcodeCount	History	The maximum number of barcodes read.
Status	Inspection	This status of the most recent inspection either Pass, Fail, or Idle (no triggers).
FrameNumber	Inspection	The most recent inspection frame number
ExecutionTime	Inspection	The most recent inspection execution time in msec.
ReadNoRead	Inspection	The barcode decoder statsus either read or barcode not found.
CompareData	BCR_INPUT	The compare data string. This string must start and end with the double quote character.
CompareMask	BCR_INPUT	The compare string mask in binary format; that is, masked characters are indicated by a "1" and unmasked characters are "0." Note that the mask character string must match the length of the compare string.
Count	BCR_RESULT	The total number of barcodes found in the last inspection.
Data	BCR_RESULT	The barcode data strings that were read in the last inspection. Each string starts and ends with the double quote character. Multiple strings are seperated by the field delimiter
Type	BCR_RESULT	The type(s) of barcodes read in the last inspection. Multiple values are seperated by the field delimiter.
MaxPercentMatch	MATCH_RESULT	(New Item Please Add Description Here)

4.3.3 Set Commands

Set commands set some group item on the sensor (for example, set compare data for the BCR).

Command	Group	Item	Value
Set	BCR_INPUT	CompareData	<VALUE>
Set	BCR_INPUT	CompareMask	<MASK >
Set	Trigger	Mode	ContinuousScan
Set	Trigger	Mode	ExternalSingle
Set	Trigger	Mode	ExternalGated
Set	Trigger	Mode	Command

4.3.4 Command Channel Response Frames

The iVu responds to all request frames with one or two responses depending on the type of command.

Do commands

All do commands are followed by one response that identifies the command status. For example:

```
>>  do trigger\x0D\x0A
<<  OK\x0D\x0A
```

Get commands

All get commands are followed by two responses: the first identifies the status of the command, and the second contains the retrieved information. For example:

```
>>  get bcr_input comparedata\x0D\x0A
<<  OK\x0D\x0A
<<  "012345ABCDEF"\x0D\x0A
```

Set commands

All set commands are followed by one response that identifies the command status. For example:

```
>>  set bcr_input comparedata "012345ABCDEF"\x0D\x0A
<<  OK\x0D\x0A
```

4.3.5 Command Channel Command Status

The command status is either OK or ERROR. If OK, then the command has fully and successfully completed. If an error is returned it is in the form *ERROR nnnnn_ERROR_IDENTIFIER* (for example ERROR 10001_COMMAND_NOT_RECOGNIZED). Refer to *Command Channel Error Codes* on page 19 for a list of errors.

5 Examples

5.1 Conventions Used for Examples

There are a number of command channel examples included here, and the following are the conventions used in the examples:

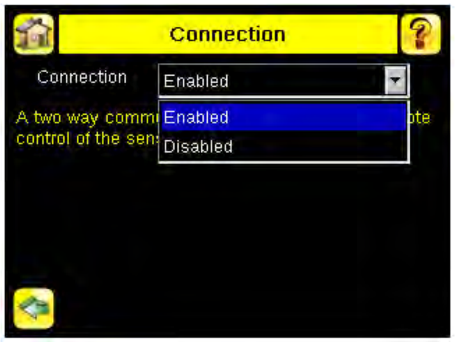
- All examples use <CR><LF> for the end-of-frame delimiter, and this delimiter is always denoted in hex (\x0D\x0A) since that is what is displayed in the iVu logs and, for example, the Port Status screen.
- All commands are in bold text.
- For each example, a command request to the iVu sensor is prefaced with a >>, and a command response frame from the iVu sensor is prefaced by a << as shown below. These are only used to make the documentation clearer.

```
>>  get info companyname\x0D\x0A
<<  OK\x0D\x0A
<<  "Banner Engineering Corp."\x0D\x0A
```

5.2 How to Trigger the Sensor and Retrieve Barcode Data using the Command Channel

To trigger the sensor and retrieve barcode data, do the following:

1. Main Menu > System > Communications > Command Channel > Connection and select Enabled.



2. Set Trigger to Command. Go to the Main Menu > Imager > Trigger screen, and from the drop-down select Command
3. Issue a trigger command as follows:

```
>>  do trigger\x0D\x0A
<<  OK\x0D\x0A
```

4. Check that the inspection passed.

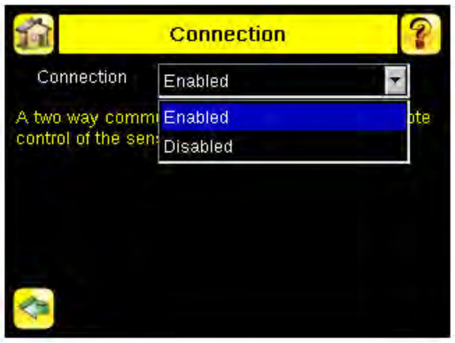
```
>>  get inspection status\x0D\x0A
<<  OK\x0D\x0A
<<  Pass\x0D\x0A
```

5. Get the barcode data read by the iVu sensor.

```
>>  get bcr_result\x0D\x0A
<<  OK\x0D\x0A
<<  "0043000011201"x0D\x0A
```

5.3 How to Modify Barcode Compare Data Using the Command Channel

1. Main Menu > System > Communications > Command Channel > Connection and select Enabled.



2. Set Trigger to Command. Go to the Main Menu > Imager > Trigger screen, and from the drop-down select Command.
3. Set the compare data.

```
>>  set bcr_input comparedata "0043000011201"\x0D\x0A
<<  OK\x0D\x0A
```

4. Trigger the sensor.

```
>>  do trigger\x0D\x0A
<<  OK\x0D\x0A
```

5. Check that the inspection passed.

```
>>  get inspection status\x0D\x0A
<<  OK\x0D\x0A
<<  Pass\x0D\x0A
```

6. Get the barcode data read by the iVu sensor.

```
>>  get bcr_result data\x0D\x0A
<<  OK\x0D\x0A
<<  "0043000011201"\x0D\x0A
```

6 Command Channel Reference

6.1 BCR_INPUT Command Group

Command	Group	Item	Description
get	bcr_input	comparedata	Returns the compare string.
get	bcr_input	comparemask	Returns the compare string mask. Masked characters are indicated by a "1" and unmasked characters are "0." Note that the mask character string must match the length of the compare string.
set	bcr_input	comparedata	Sets the compare string.
set	bcr_input	comparemask	Sets the compare string mask. Masked characters need indicated by a "1" and unmasked characters are "0, " and the mask character string must match the length of the compare string.

Examples:

```
>>  get bcr_input comparedata\x0D\x0A
<<  OK\x0D\x0A
<<  "0043000111201"\x0D\x0A
```

```
>>  get bcr_input comparemask\x0D\x0A
<<  OK\x0D\x0A
<<  "1111000000000"\x0D\x0A
```

```
>>  set  bcr_input comparedata "0043000111201"\x0D\x0A
<<  OK\x0D\x0A
```

```
>>  set bcr_input comparemask "1111000000000"\x0D\x0A
<<  OK\x0D\x0A
```

6.2 BCR_RESULT Command Group

Command	Group	Item	Description
get	bcr_result	count	Returns the number of barcodes found
get	bcr_result	data	Returns the barcode data that the iVu sensor read.
get	bcr_result	type	Returns the type of the barcode read. Multiple items are separated by a field delimiter.

Examples:

```
>>  get bcr_result count\x0D\x0A
<<  OK\x0D\x0A
<<  1\x0D\x0A
```

```
>>  get bcr_result data\x0D\x0A
<<  OK\x0D\x0A
<<  "0043000011201"\x0D\x0A
```

```
>>  get bcr_result type\x0D\x0A
<<  OK\x0D\x0A
<<  EAN13\x0D\x0A
```

6.3 History Command Group

Command	Group	Item	Description
get	history	passed	Returns the number of passed inspections.
get	history	failed	Returns the number of failed inspections.
get	history	missedtriggers	Returns the number of missed triggers.
get	history	startframenumber	Returns the start frame number.
get	history	endframenumber	Returns the end frame number.
get	history	mininspectiontime	Returns the minimum elapsed time of the inspection.
get	history	maxinspectiontime	Returns the maximum elapsed time of the inspection.
get	history	minbarcodecount	Returns the minimum number of barcodes read.
get	history	maxbarcodecount	Returns the maximum number of barcodes read.
do	history	clear	Clears all history fields (for example pass, fail, etc.).

Examples:

```
>>  get history passed\x0D\x0A
<<  OK\x0D\x0A
<<  13\x0D\x0A
```

```
>>  get history startframenumber\x0D\x0A
<<  OK\x0D\x0A
<<  3\x0D\x0A
```

```
>>  get minbarcodecount\x0D\x0A
<<  OK\x0D\x0A
<<  1\x0D\x0A
```

```
>>  do history clear\x0D\x0A
<<  OK\x0D\x0A
```

6.4 Info Command Group

Command	Group	Item	Description
get	info	companyname	Returns the company name.

Command	Group	Item	Description
get	info	modelnumber	Returns the sensor model number.
get	info	firmwareversion	Returns the sensor firmware version.
get	info	serialnumber	Returns the sensor serial number.
get	info	bootnumber	Returns the number of sensor bootups.
get	info	name	Returns the sensor name.
get	info	uptimer	Returns the elapsed time the sensor has been running in the format hh:mm:ss:ms.
get	info	hourcount	Returns the number of hours the sensor has been running.
get	info	remoteconnected	Returns the whether a remote display is connected as a boolean value (true or false).
get	info	remoteserialnumber	Returns the the serial number of the remote display.
get	info	remotemodelnumber	Returns the model number of the remote display.

Examples:

```
>>   get info companyname\x0D\x0A
<<   OK\x0D\x0A
<<   "Banner Engineering Corp."\x0D\x0A
```

```
>>   get info bootnumber\x0D\x0A
<<   OK\x0D\x0A
<<   42\x0D\x0A
```

```
>>   get info uptimer\x0D\x0A
<<   OK\x0D\x0A
<<   4:42:42:324\x0D\x0A
```

6.5 Inspection Command Group

Command	Group	Item	Description
get	inspection	status	Returns either Pass, Fail, or Idle.
get	inspection	readnoread	Returns either Read (found) or NoRead (not found).
get	inspection	framenumber	Returns the frame number.
get	inspection	executiontime	Returns the inspection execution time.

Examples:

```
>>   get inspection status\x0D\x0A
<<   OK\x0D\x0A
<<   "Fail"\x0D\x0A
```

```
>>   get inspection readnoread\x0D\x0A
<<   OK\x0D\x0A
<<   Read\x0D\x0A
```

```
>>   get inspection executiontime\x0D\x0A
<<   OK\x0D\x0A
<<   37.739\x0D\x0A
```

6.6 Status Command Group

Command	Group	Item	Description
get	status	ready	Returns either True (ready for trigger) or False.
get	status	systemerror	Returns either True or False.
do	status	clearsystemerror	Resets SystemError to False.

Examples:

```
>>  get status ready\x0D\x0A
<<  OK\x0D\x0A
<<  True\x0D\x0A
```

```
>>  get status systemerror\x0D\x0A
<<  OK\x0D\x0A
<<  False\x0D\x0A
```

```
>>  do status clearsystemerror\x0D\x0A
<<  OK\x0D\x0A
```

6.7 System Command Group

Command	Group	Item	Description
do	system	save	Saves inspection and configuration parameters. Blocks until finished. Should be used sparingly.
do	system	reboot	Reboots the sensor. Pre-empts all commands except save.

```
>>  do system save\x0D\x0A
<<  OK\x0D\x0A
```

6.8 Teach Command

Sets the sensor to teach on the next trigger

Example:

```
>>  do teach\x0D\x0A
<<  OK\x0D\x0A
```

6.9 Trigger Command Group

Command	Group	Item	Description
set	trigger	mode	Sets trigger mode to one of the valid trigger modes for the sensor: ContinuousScan, External, ExternalGated, or Command.

Command	Group	Item	Description
get	trigger	mode	Returns the trigger mode.
do	trigger		Initiates a single trigger. The sensor does not transmit a response until the sensor has completed the action.
do	trigger	gated	Initiates gated triggering. The sensor does not transmit a response until the sensor has completed the action.
do	trigger	abortgated	Aborts gated triggering. The sensor does not transmit a response until the sensor has completed the action, then it will respond with two responses (one from the previous gated trigger and one for the abort).

Examples:

```
>>  set trigger mode command\x0D\x0A
<<  OK\x0D\x0A
```

```
>>  get trigger mode\x0D\x0A
<<  OK\x0D\x0A
<<  Command\x0D\x0A
```

```
>>  do trigger\x0D\x0A
<<  OK\x0D\x0A
```

The following example shows a gated trigger that is taking too long so an do trigger abortgated command is executed.

```
>>  do trigger gated\x0D\x0A
>>  do trigger abortgated\x0D\x0A
<<  OK\x0D\x0A
<<  OK\x0D\x0A
```

6.10 Command Channel Error Codes

Table 1: BCR Command Channel Error Codes

Numeric ID	Text ID	Description
00000	SUCCESS	Command processed successfully
10000	EMPTY_FRAME_RECEIVED	Indicates that the request was empty. The command channel requires a command, any arguments, and an end-of-frame delimiter.
10001	COMMAND_NOT_RECOGNIZED	The command specified is not recognized
10100	GROUP_MISSING	A Group ID must be specified immediately after the command
10101	GROUP_NOT_FOUND	The specified Group ID is invalid / unknown
10102	GROUP_ITEM_MISSING	A Group Item ID must be specified immediately after the Group ID
10103	GROUP_ITEM_NOT_FOUND	The specified Group Item ID is invalid / unknown
10152	NOT_READABLE	Attempt to get a value that is not readable
10153	NOT_WRITEABLE	Attempt to set a value that is not writeable
10250	NOT_A_METHOD	Method ID specified is not a method
10251	WRONG_ARGUMENT_COUNT	Total method arguments specified do not match method
10252	COMMAND_NOT_FINISHED	Attempt to issue command when a previous command has not finished
10300	INVALID_ARGUMENT_TYPE	Item ID specified must be a item (not a group or method)
10301	DATA_VALUE_MISSING	Command missing item's data value
10350	ARGUMENTS_DETECTED	Get command received with unneeded arguments
10351	INVALID_ARGUMENT_TYPE	Item ID specified must be a item (not a group or method)

Numeric ID	Text ID	Description
10340	MINIMUM_VALUE_EXCEEDED	New item value is below the minimum
10341	MAXIMUM_VALUE_EXCEEDED	New items value is above the maximum
10500	DATA_SET_EMPTY	Data export operation returned no results.
10900	SENSOR_NOT_READY	Command specified requires sensor to be in the READY state.
10920	SENSOR_TYPE_NOT_ACTIVE	Command specified belongs to a different sensor type.
15000	VALUE_INVALID	Text value is invalid / unknown
15050	VALUE_INVALID	Text value is invalid - expecting True or False
15100	STRING_TOO_LONG	String value specified exceeds maximum allowable length
20000	BARCODE_TIMEOUT	Attempt to obtain Barcode result data when decoder has timed out
20001	NO_BARCODES_FOUND	Attempt to obtain Barcode result data when no barcodes were found
20002	COMPARE_DATA_DISABLED	Operation requires Barcode compare to be enabled
20003	COMPARE_MASK_INVALID	Compare mask invalid. Expecting string of 1's and 0's with length equal to compare data string
20004	NUMBER_TO_FIND_NOT_ONE	Barcode number to find must be set to one for this operation.
80000	REMOTE_DISPLAY_NOT_CONNECTED	Remote Display must be connected to obtain this value
80001	REMOTE_DISPLAY_NOT_SUPPORTED	This sensor does not have Remote Display capability
80100	COMMAND_MODE_EXPECTED	The Trigger Mode must be set to "Command" perform this operation
80101	COMMAND_TIMED_OUT	The command timed out before finishing
80102	TRIGGER_REQUIRED	Access to the specified data requires a triggered inspection
80103	TRIGGER_NOT_GATED	Command requires a active Gated Trigger
80150	COMMAND_TIMED_OUT	The command timed out before finishing
80200	SYSTEM_ERROR_NOT_ACTIVE	The System Error must be active to execute this command

Index